

[illegible]

GRAPHICS EDITOR SYSTEM

POP OVER MENUS

80 COLUMN OUTPUT ON THE C128

COMAL 2.0 KEYWORDS CHART

RAM DISK

DISK SLEEVE DIRECTORIES

PACKAGE DE-LINK PROGRAM

page 65

OKIDATA GRAPHICS DUMP

page 56

BEGINNERS

- 1 - Editors Disk
- 4 - COMAL Comments - Sol Katz
- 22 - Disk Directory Tips
- 38,62,72 - Questions and Answers
- 69 - Educator Needs - Jim Ventola
- 79 - How to Type in COMAL Programs
- 82 - Filename Conventions
- 82 - COMAL 0.14 Error Messages

PROGRAMMING

- 5 - Live Menu - Christopher J. Abissi
- 18 - Your RUN/POP Key - Len Lindsay
- 24 - COMAL Clinic
- 26 - C128 80 Column Screen - David Stidolph
- 37 - Statistics - David Powell
- 55 - 2.0 Auto Save - Marvin Cook
- 56 - OkiData Graphics Dump - Terry Ricketts
- 57 - Graphics Editor System - Colin Thompson

ADVANCED USERS

- 16 - Fix Disk Errors - David Stidolph
- 38 - 2.0 Modem Update - David Stidolph
- 66 - Differentiation - Tom Kuiper
- 67 - FFT - Tom Kuiper

2.0 PACKAGES

- 63 - Text Package - Dick Klingens
- 65 - DeLINK a Package - Dick Klingens

GENERAL

- 5 - Type Old English - Phyrne Bacon
- 6 - GreyMat - Joel Rea
- 8,68 - Bug Fixes
- 76 - Letters

BOOKS

- 12 - Best Selling Books
- 13 - Book Review - Starting With COMAL

REFERENCE

- 10 - COMAL Implementations
- 14 - Control Key Values - Larry Winckles
- 30 - COMAL 2.0 Keywords - Daniel W Parish
- 40 - Disk Directories

ADVERTISERS

- 3 - Transactor
- 9 - TPUG
- 23 - ICCE - Computing Teacher
- 39 - Aquarian Software
- 71 - SPARCUG
- 78 - Midnite Gazette
- 78 - CASE
- 81 - United States Commodore Council

EDITOR

Len Lindsay

ASSISTANTS

Denise Bernstein
Maria Lindsay
David Stidolph
Geoffrey Turney

ART

G. Raymond Eddy

CONTRIBUTORS

Christopher Abissi
Phyrne Bacon
Anthony Conca
Marvin Cook
John Eldredge
Peter Gilbert
Dennis Johnson
Sol Katz
K Keller
Dick Klingens
Tom Kuiper
Len Lindsay
Rhianon Lindsay
Ian MacPhedran
Bob McCauley
Rodney McDaniel
Martin Page
Daniel Parish
David Powell
Joel Rea
Terry Ricketts
Kevin Ryan
W Staneski
David Stidolph
Colin Thompson
Jim Ventola
Larry Winckles
Paul Winslow

PUBLISHER

COMAL Users Group,
USA, Limited
6041 Monona Drive
Madison, WI 53716
608-222-4432

From the Editor's Disk

We agree that COMAL is preferred over other programming languages. Now it is official from Commodore USA! COMAL is the third most popular language used on Commodore computers. BASIC & ML are higher only because the computer comes with them. Thus COMAL is the **number 1 language of choice** (people don't choose BASIC, they merely accept it).

Good news. If you have any of our COMAL disks, you now can have it's entire directory pasted onto the front of your disk sleeve. The directories of almost all our COMAL disks are printed in this issue - a full 15 pages of directories! Each is printed in just the right width to fit perfectly on the front of a disk sleeve. Just photocopy the page and cut out the sleeve overlays you need.

Even more good news. The matching *Today Disk #11* contains a complete **graphics editor system** (even 2.0 people will now have an excuse to use 0.14 again). Each program depends on the others in the system. The programs cannot be used separately. But, if you list the programs, you will probably find **many useful procedures**. Create your own procedure library. List the ones you like to disk for future use. If, for example, you wish to pull out a procedure that is at lines 350-420, issue this command:

list 350-420 "all'mine.proc"

Almost half of our newsletter subscribers also subscribe to the *Today Disks*! Because of this we can publish special systems as part of the newsletter / disk set (such as the Graphics Editor of this issue). And while the age of typing in programs may be dead, we will continue to list our programs.

Everything announced last issue is now available, including the *Packages Library* book/disk set (already on the top 5 best sellers list). It includes information on using 17 different packages for the 2.0 Cartridge. The ready to LINK packages are on the disk as well as the full source code (when available). As a bonus, the disk also includes a Smooth Scrolling Editor with full screen editing (lines kind of glide across the screen). Even Commodore wanted a copy of it.

Next issue we plan to announce four more COMAL books. One is a superb full size text book which has a down to earth way of looking at learning programming, designed with the American student in mind! Complete with objectives. Also coming soon are a tutorial for 2.0 Graphics, a beginners guide to COMAL, and a 2.0 COMAL introduction.

While most "big" magazines make a big deal out of one feature program, we provide many feature programs. This issue you get the 80 column screen on a C128 from COMAL breakthru, a POP OVER menu system, complete graphics editing system, and more. Next issue we plan to include *Sideways* (print Multi plan spread sheets sideways), a double column file printer, a package to transfer a user defined font from the computer to your printer, and a C128 package.

Other magazines may be hurting, but have no fear. We're fine. January 1986 orders were better than both November or December 1985, and we have a fine supply of programs and articles that are already being put together for our next issue. (Sorry about the delay on this issue, but we lost our associate editor in December - the job is still open). ■

COMALites Unite!

April 26 and 27 is a Commodore Show in Nashville, Tennessee. Stop by the Transactor booth to get a copy of our new 24 page COMAL Info Booklet. You may not realize that the *Transactor* is one of the oldest Commodore magazines. They provide the technical information you need. I've subscribed to Transactor for years and recommend it.

In September we hope to be at the Commodore Show now being planned for Los Angeles. More details later.

COMAL is now supported on several national On-Line networks. Meet Captain COMAL (Captain C) on Playnet every first Thursday of the month or on Quantum Link every second Thursday. Meeting starts at 10pm Eastern Time. Tuesday nite is the time to be on People Link (leave EMAIL to ICONOCLAST). CompuServe has a Beyond BASIC section with ongoing COMAL information. Delphi is now incorporating COMAL into both the Commodore Flagship and TPUG sections.

Questions? Get them answered within a few days (allow a week) on Quantum Link. There is a special COMAL Questions and Answers board. Captain COMAL tries to stop in at least once a week and post answers to all questions left on the board. Other COMALites are welcome to respond to the questions as well. To get there go to the Commodore Information Network (CIN). Then choose **Meet The Press**. Next pick **COMAL Today**. Finally, choose **Q & A**. You can read all the questions and their answers. Some of your questions may already be answered!

Wonder where all the neat COMAL programs come from? From people like you! This is your newsletter, and your *Today Disks*. If you develop interesting COMAL

programs, send them in. In exchange we will send you a *User Group Disk* of your choice (already 11 to choose from). Articles about any COMAL related topic are always welcome. Please send the text as disk files (in PaperClip, WordPro, Paperback writer, or EasyScript format if possible).

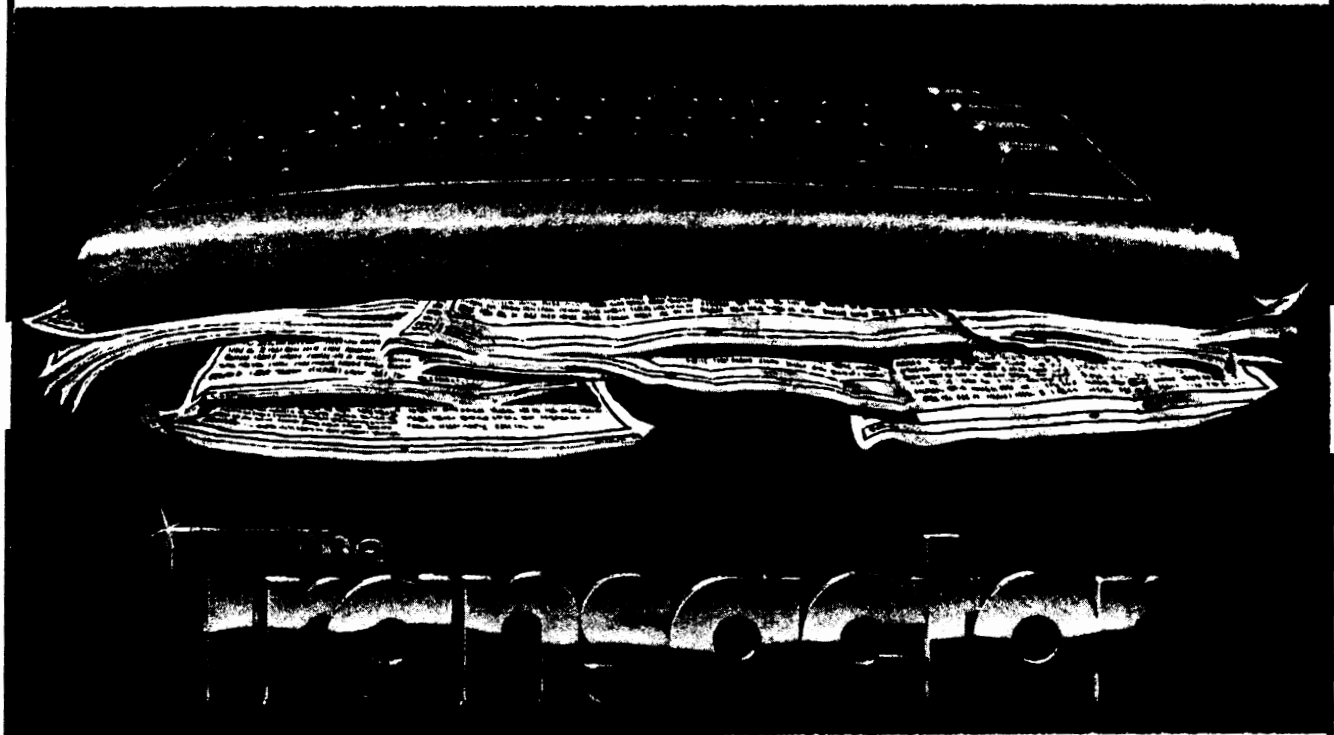
If you ever get a defective disk from us, return it within 30 days and get a free replacement. You also can get an extra "backup" copy of any disk you order from us for \$5 (in case you have trouble making your own backup copy). We now are making our own copies of the matching disk to the *Cartridge Tutorial Binder*. Too many disks we received from Commodore had problems.

We encourage you to write to the "big" Commodore magazines. Can you afford to subscribe to a magazine that doesn't cover COMAL? If you read *Ahoy*, send in solutions to their *Commodares* problems. They are usually easier to solve in COMAL. If you read *RUN*, they now will officially accept COMAL *Magic* tips! Send them your hints and tips today! Finally, *Ahoy* has decided to include COMAL programs on it's monthly disks. All we have to do is send the programs to them. Sounds promising.

Our free subscription to schools continues (only one per school, USA only). We also continue to exchange newsletters with other User Groups (put us on your group mailing list and we add you to ours) - also USA only.

Finally, we plan on releasing an **ICON Disk** in April. Some of the pictures that have been submitted to us for it are printed in this issue. If you have any custom icons please send them in.

Expand Past Maximum Capacity!



The Tech/News Journal For Commodore Computers

At better book stores everywhere! Or 6 issues delivered to your door for just \$15.00

That's 29% off the newstand price! (Overseas \$21 U.S. Air Mail \$40 U.S.)

The Transactor, 500 Steeles Ave. Milton, Ontario. L9T 3P7. 416 878-8438

Also check out The Transactor Disk; every program from each issue, in order as they appear

and The Complete Commodore Inner Space Anthology; over 2.5 million characters of reference information exclusively.

Included are memory maps for BASIC, COMAL, CBM disk drives, BBS numbers, machine language charts, Kernel routine summaries, printer commands, recording formats, I/O port maps, port pinouts, audio control tables, video reference charts, keyword values, commands for common software packages, MLM and assembler commands, and there's more!

To us, expansion knows no limits!

COMAL Comments

by Sol Katz

Perhaps the most unusual command structure in COMAL (from a BASIC programmer's point of view) is the CASE structure. The closest thing in BASIC is the ON .. GOTO or ON .. GOSUB.

The CASE structure allows multiple branches in a program, depending on the value of the controlling variable. Unlike BASIC, the variable can be either numeric or string, and there is no need for the choices to be in any order. The keywords that make up the CASE structure are underlined:

```
case variable of  
when value  
    // statements go here  
when value  
    // statements go here  
otherwise  
    // statements go here  
endcase
```

The controlling variable comes immediately after the keyword CASE. Each choice is prefixed with the keyword WHEN. The OTHERWISE section statements are executed if the variable has a value that was not one of the choices. ENDCASE ends the structure.

As a bonus, more than one value for a variable can be part of the WHEN clause. For example:

```
input "enter a 1 digit number:":number  
case number of  
when 7,5,1,9,3 //order is not important  
    print "the number is odd"  
otherwise  
    print "the number is even"  
endcase
```

When using string variables, the values must be in quotes. For example:

```
dim letter$ of 1  
input "Type a capital letter: ":letter$  
case letter$ of  
when "A","E","I","O","U"  
    print "This is a vowel"  
when "Y"  
    print "This is sometimes a vowel"  
otherwise  
    clear'screen  
    print "This is a consonant"  
    print "This shows another statement"  
endcase  
//  
proc clear'screen  
    print chr$(147),  
endproc clear'screen
```

If you haven't guessed by now, // is the COMAL equivalent to REM in BASIC. It indicates a comment. *[You also can use an exclamation point (!) which COMAL will convert into // for you.]*

The last bonus of the case structure is that string variables are not limited to one letter. For example:

```
dim month$ of 3  
input "Enter name of a month:":month$  
case month$ of  
when "Jan","jan","Feb","feb","Mar","mar"  
    print "It's winter and I'm cold"  
when "Apr","apr","May","may","Jun","jun"  
    print "The flowers are blooming"  
when "Jul","jul","Aug","aug","Sep","sep"  
    print "Sure is hot"  
when "Oct","oct","Nov","nov","Dec","dec"  
    print "Pretty colors"  
otherwise  
    print "Either you're a wise guy or"  
    print "you don't know what a month is"  
    print "- I don't know that month."  
endcase
```

More ►

Live Menu

[Notice in the above example that MONTH\$ is dimensioned to 3 characters. Thus, the user could type in the full name of the month (such as January) and the program would still work properly, since COMAL would only remember the first 3 letters of what was typed in. Also, note that the first letter of the month name can be capitalized or not.]

In most of my examples I have used PRINT statements, but you can use any valid COMAL statements and any number of statements as part of a WHEN or OTHERWISE clause. As another bonus, no matter how you enter the program, COMAL will format it to appear as it does in these examples, and if you make syntax errors as you enter a program, COMAL will tell you, then and there. Now aren't you glad you switched to COMAL? If you haven't, remember that copies of the language are available in our club library.

[COMAL Users Group USA Ltd has over 50 different COMAL disks now. See order form at the back of this newsletter. This article was reprinted from CUFLINK, July 1985] ■

TYPE IN OLD ENGLISH

by Phyrne Bacon

This is a new font - only BIG! The directions on how to type in Old English appear when you run the program, followed by a quote from *Through the Looking Glass*. Then every valid character that you type is echoed on the screen in Old English lettering. Press *f7* to transfer the text to the graphics screen (it can then be dumped to your printer). Press *f1* to exit. A version for both COMAL 0.14 and 2.0 are on the back side of *Today Disk #11* along with the Old English font. ■

by Christopher J. Abissi, M.D. and friends

Utility programs frequently employ menus to direct the user to the different portions of a program. A menu with a number of options can take several seconds to print on the screen. Under optimal conditions the C64 takes between 3 and 4 seconds to fill its screen. This does not pose a problem to the novice user. However, after one has become familiar with the options it can become tiresome to repeatedly wait for the menu to print.

In BBS programs or terminal programs the screen printing is even slower, and it can take 26 seconds to completely fill the C64 screen. Permitting faster menu selection decreases the time the user is on-line (lower connect charges).

I have written a "live" menu demo program to help with this problem. With this type of menu the user can make a choice as soon as the menu begins printing and go straight to the option. A second menu can be accessed from the first. Following similar structures even more can be linked. The menu information is stored in DATA statements at the end of the program permitting it to be customized as desired. To handle the options a CASE structure was used with an IF structure nested within to handle the second menu. The COMAL language and the comments that have been included make the program largely self-explanatory.

Enjoy the program and feel free to use the menu procedure in your own programs. We may have to wait for our disk drives, but COMAL users no longer have to wait for their menus! *[This 2.0 program is on Today Disk #11]* ■

GreyMat - Think Into Your C64

by Joel Rea

We interrupt this issue of *COMAL Today* to bring you an announcement of a most important new product. Forget paddles, joysticks, trackballs, lightpens, touch pads, touch screens, mice, foot-mice and voice recognition. The ultimate in user-friendly input is here! From Cerebrionics Hardware Enterprises comes the all-new GreyMAT Brainwave Scanning Interface! This amazing device actually digitizes input from your own "Grey MATter", then compares it with an up to 64-thought "Psychabulary" file. The driver software then returns the closest match to the calling program. Though not completely perfected, GreyMAT is still the most important advance in computer science since the invention of binary numbers!

It's also malarkey! The program "greymat" on this disk is actually a mentallist-type parlor trick. When RUN, this program pretends to read in the GreyMAT driver routines and a 52 thought Bridge-Card Psychabulary file. Balderdash! What the program is actually doing is scanning 2 junk files on the disk. When done, it will display a fake "Copyright" notice, show a message asking the user to think of a playing card and re-RUN the program, then it exits to COMAL 2.0's "Ready" prompt.

More baloney! It just *looks* like it's in COMAL! You can type in and execute most of the COMAL 2.0 commands (such as DIR, CAT, LIST, FIND, DISPLAY, etc.), use full-screen editing, change cursor color as well as use CTRL V, W, X and Y to change the background and border colors, and even get hardcopies of the text and/or the graphics screen via CTRL D and H. (CTRL A, B, F, K and L are not

simulated.) You can even execute immediate mode COMAL statements! Don't generate an error, it will cause GreyMAT to "crash out". Just re-RUN if this occurs (you will hear 2 error beeps).

When someone types in a RUN command (without filename, of course!), the GreyMAT program will claim to be reading the non-existent GreyMAT device, then analyzing the data it never got. It will then state that the card you were thinking of was some card it generated randomly.

What's the trick, then? Type the "R" key as if you were going to do a "RUN" command (the cursor must be at the full left-hand margin). Now, instead of typing "U", type "5". What's this? The computer echoed "U" instead of "5"! Now type "C" in place of "N". Hmmm... the computer echoes "N". The word "RUN" is on our screen (even though we typed in R5C). Press <return>. The program goes through its normal fake read/analyze routine, then states that the card you were thinking of was a Five of Clubs! Hmmm... "5" of "C"lubs? Let's try another. Type "RQH", which will echo as "RUN" and press <return>. The computer claims you were thinking of a Queen of Hearts! Now you see the pattern. Typing "R" followed by the desired Rank and Suit of the card will cause the computer to claim that you were thinking of that card! The poor "sucker" you are pulling this on will think you just typed "RUN", since that is what is on the screen, and since the other regular COMAL commands work.

If the "sucker" wants to be at the keyboard, just let him. Naturally, he will type "RUN", and the computer will choose a random card. Of course, that

More ►

means there is a 51-in-52 chance that it wasn't the right card. Just say that the "Psychabulary" file was made by you with the "training" program and that it takes 3 hours to create a file for another person, or that the person was thinking about his typing and not the card when he actually pressed the <return> key, etc.

If you *really* want to carry this trick/hoax/ fraud/ April Fool's joke to its logical conclusion, just build yourself a "GreyMAT" device! All you need is a cloth jogger's-type headband or a ski cap or a baseball cap or some such, and a joystick wire and connector. Just stick the wire end of the joystick wire into the headware somewhere, and plug the other end into one of the Game Ports. If done right, it looks real enough to fool Jesse Knight himself! (I know, I pulled it on him, Len, Borge and others at MARCA Fair!)

The most convincing way to do the trick is to obtain a deck of marked cards (DeLand's Automatic Deck, available at magic shops, works fine!). You can then have the "sucker" shuffle and cut the deck and place it face down beside the computer. You then read the back of the top card, then type "RUN" (not really, of course!), then pick up the card (pretending that only now do you know what it is), then press `<return>` while pretending to concentrate real hard on the card. I also suggest you use a TV, or an SX-64 connected to an external monitor, or at least have your 64 some distance away from the monitor, so the patsies' attention is drawn away from what you are *really* typing!

Well, since I embarrassed poor Jesse Knight a paragraph or two ago, I will

1/lensctl1/mccauley-bugs/MISSING KE


point out that this program would not have been possible without his article entitled "*Batch Files from Memory*" in *COMAL Today* #7, pages 32 & 33. How else do you think I can simulate the COMAL editor from within a COMAL program?

One last note. I suggest before showing some poor shmuck the GreyMAT Card Demo, that you let him read the first paragraph of these instructions! [photo copy it and black out rest of page.] Then, after you have thoroughly bam-boozled him with the trick, instead of just telling or showing him how it really works, just let him read the REST of the story! Watch him do a slow burn! Heh heh heh...

[Ed Note: Don't let your friends read this COMAL Today until you've tried this program on them. It reminds me of "Chinese Numbers". Building your own GreyMat device is quite easy (since it doesn't really do anything) and makes your "show" quite convincing. You may wish to turn off the computer after running this program, though Joel assures us that it is not necessary.]

MISSING KEYWORDS

Question: Did I miss something, or did the *COMAL-80 Tutorial Binder* omit the description for SETSCORE in the Sound package section? I find references to it in the narrative and in the summary on page 191. On the other hand, The *Cartridge Graphics and Sound* book had the SETSCORE procedure but omitted talking about either FREQUENCY or PLAYSCORE. - Bob McCauley, APO, NY

Answer: Ooops. 

Bug Fixes

HANDBOOK FIX

Page 220 of the *COMAL Handbook* should be appended to allow for the C64 COMAL 2.0 Cartridge, whose control location is at \$C7D8 rather than at \$24B. Also note a change in the meaning of the first two bits:

Bit 0: 1=normal C64 mode
0=convert control codes to
 ""<value>""

Bit 1: 1=normal C64 mode
0=ignore quote mode & insert mode

The default value of these bits in the control location are:

Decimal 128, Binary %10000000

FIX FOR SOUNDEX

Martin Page sent us this note about the SOUNDEX program we printed in *COMAL Today* #8, page 22:

I have just inserted the SOUNDEX function into a mailing list program for our local wildlife films but was frustrated by substring errors. The solution did not occur to me rapidly, although it is very simple. Names with non-alphabetic characters such as O'Connor or double-barrelled names with a hyphen cause the problem. To discard these characters add the apostrophe and hyphen in the line:

```
if not (name$(i#) in "AEHIUWY'-' then
```

Perhaps this could help other programmers using the SOUNDEX function (or procedure in COMAL 0.14).

DIR'PRINTER3 FIX

There is a bug in the *dir'printer3* program on the *Best Of COMAL* disk. Line 1270 should remove the '3 on the end of the procedure name:

From: 1270 print'heading'3
To: 1270 print'heading

WORD GAME FIX

Option 5 of the *Word Game* in *COMAL Today* #8 doesn't work. Peter Gilbert has provided this fix:

```
WHEN "5"
FOR n:=1 TO LEN(text$) DO
  IF text$(n) IN punc$ OR text$(n)=" "
  THEN // wrap line
    guess'text$(n):=text$(n)
  ELSE
    guess'text$(n):="-"
  ENDIF
ENDFOR n
//
//
n:=1
LOOP
  REPEAT
    IF text$(n) IN alf$ THEN
      guess'text$(n):=text$(n)
    ENDIF
    n:=1
  UNTIL text$(n)=" " OR n=LEN(text$)
  EXIT WHEN n>=LEN(text$)
  REPEAT
    IF text$(n) IN alf$ THEN
      guess'text$(n):="-"
    ENDIF
    n:=1
  UNTIL text$(n)=" " OR n=LEN(text$)
  EXIT WHEN n>=LEN(text$)
  //
ENDLOOP ■
```

TPUG

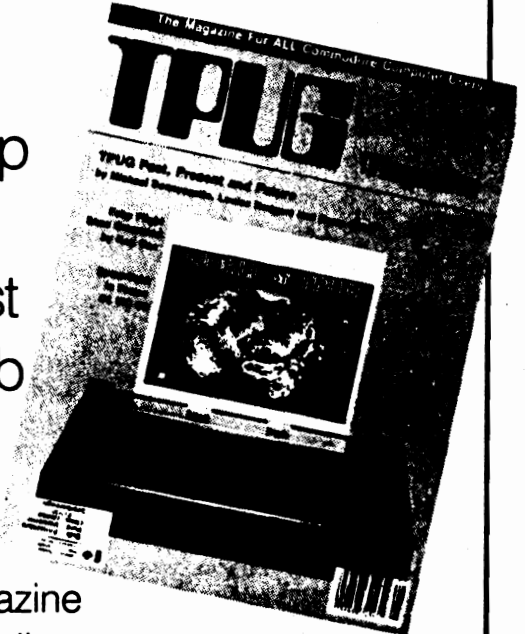
MORE THAN JUST ANOTHER COMPUTER MAGAZINE!



TPUG
101 Duncan Mill
Suite CT
Don Mills, Ontario
CANADA
M3B 1Z3

A membership
in the
world's largest
computer club
will provide
you with:

- 10 issues of TPUG magazine
- Advice from experts like Jim Butterfield and Elizabeth Deal
- Access to our huge library of public domain software
- Access to our online help services
- All for only \$25/yr.



JOIN US NOW!!

I want to join TPUG as an associate member.

Name _____	Home phone (____) _____
Address _____	Work phone (____) _____
City/Town _____	<input type="checkbox"/> Cheque enclosed
Prov/State _____	<input type="checkbox"/> Money order enclosed
Postal/Zip Code _____ Country _____	<input type="checkbox"/> Visa <input type="checkbox"/> MasterCard
My computer equipment is: _____	Card# _____
_____	Expiry _____
_____	Signature _____

COMAL Implementations

As of September 1984
(updated February 1986)

*Compiled by: Kevin Ryan, Secretary of
the COMAL Standards Group, Department of
Computer Science, Trinity College,
Dublin 2, Ireland*

AcornSoft

Betjeman House
104 Hills Rd
Cambridge CB2 1LQ
ENGLAND
Phone: +44-233-316039

Hardware:
BBC Micro A
BBC Micro B
Acorn Electron

Operating System:
BBC MOS
Electron MOS

Commodore Data

Jan Nymand
Bjerrevej 67
DK - 8700
Horsens, Denmark
Phone: +45-5-641155

Hardware:
Commodore 64
Commodore 128

Operating System:
Commodore DOS

See also UniComal

Dansk Data Elektronik

Herlev Hovegade 199
DK-2730 Herlev
Denmark
Phone: +45-2-845011

Hardware:
SPC/1
Supermax

Operating System:
Mikados
Supermax OS
Unix

IBM Denmark

Teknikerbyen 3
DK - 2830
Virum, Denmark

Hardware:
IBM PC
IBM PC XT
IBM PC AT
Zenith 151
Compaq Portable
Compaq Deskpro
Panasonic Sr. Partner
Tandy 2000
Tandy 1000
Amiga with Transformer

Operating System:
MS DOS
PC DOS

Part Number (Bestillingsnummer):
IBM COMAL-80
8132532.

Instrutek

Christiansholmsgade
DK - 8700
Horsens, Denmark
Phone: +45-5-6111100

Hardware:
PET 2001
CBM 4016
CBM 4032
CBM 8032
CBM 8096

More ►

Best Selling Books

TCD COMAL

Software Engineering Laboratory
Dept. of Computer Science
Trinity College
Dublin 2, Ireland .
Phone: +353-1-772941

Hardware:

VAX
PDP 11

Operating System:

VAX/VMS
RSTS/RSX

UniComal Aps

Jens Erik Jensen
Snogaardavej 10C
DK - 2820
Gentofte, Denmark
Phone: +45-1-651762


Hardware:

PET 2001
CBM 4016
CBM 4032
CBM 8032
CBM 8096
SUPERPET
Commodore 64
Commodore 128

Operating System:

Commodore DOS

SPECIAL NOTE:

We have an investor who would like to sponsor someone or some team to develop a COMAL for the 128K Apple IIe / IIc. This could be a major product. It should have the features of C64 COMAL 0.14 (less sprites) plus PAGE, STR\$, VAL, and GET\$. Anyone interested should contact us right away: COMAL Users Group, USA, Ltd, 6041 Monona Dr, Madison, WI 53716. 

Captain COMAL books are edging out the old favorites. *COMAL Yesterday* and *Packages Library* are new additions. Speaking of library, the book/disk set *Library of Functions and Procedures* just missed again. To give it a boost, we now include *Utilities #1* disk with each copy. Watch next issue for **THREE** new Captain COMAL books for COMAL 2.0, one a full size 300 plus page textbook!


November 1985

- #1 - COMAL From A To Z**
by Borge Christensen
- #2 - Cartridge Tutorial Binder**
by Frank Bason & Leo Hojsholt
- #3 - Cartridge Graphics and Sound**
by Captain COMAL's Friends
- #4 - COMAL Workbook**
by Gordon Shigley
- #5 - COMAL Handbook**
by Len Lindsay

December 1985

- #1 - COMAL From A To Z**
by Borge Christensen
- #2 - Cartridge Tutorial Binder**
by Frank Bason & Leo Hojsholt
- #3 - Cartridge Graphics and Sound**
by Captain COMAL's Friends
- #4 - COMAL Yesterday**
First Four COMAL TODAY Issues
- #5 - COMAL Workbook**
by Gordon Shigley

January 1986

- #1 - COMAL From A To Z**
by Borge Christensen
- #2 - Cartridge Tutorial Binder**
by Frank Bason & Leo Hojsholt
- #3 - COMAL Yesterday**
First Four COMAL TODAY Issues
- #4 - Cartridge Graphics and Sound**
by Captain COMAL's Friends
- #5 - Packages Library**
by David Stidolph 

Book Review

Starting With COMAL by Ingvar Gratte
Reviewed by Brian Grainger
Reprinted from *Independent Commodore
Products Users Group Newsletter, England*

Starting With COMAL is the fifth title on COMAL to be published in the UK. As the author says at the end, it aims to teach structured programming and top-down design. The author is a Swedish teacher who used Commodore COMAL versions predominantly throughout the book. As I was involved with reading and commenting on the manuscript it also means the book was well researched!

I should say from the outset that this book was a long time in the making, which shows a little in the various references to different COMAL versions. The original manuscript was written at the transition from 0.11 to 0.12 PET version of COMAL and I suspect it has been recently updated to cover the C64 version as well. This may lead to some confusion, as the COMAL standard was really only finalized with the latter versions. Earlier versions were steps along the way.

Quite simply, this book is in direct competition with *Beginning COMAL*, as it is a teach-yourself book on COMAL. I believe *Starting with COMAL* is better. I find it less childish and it has numerous simple exercises to reinforce the points in the book.

The book follows a natural progression from simple programs involving input and output, through conditional structures and loop structures, to use of procedures. The final chapters cover the more complex subjects of array variables, DATA statements and file handling. Random (or relative) files are

covered in the latter, which is somewhat unusual.

The text is written in such a way as to introduce some points then provide lots of simple exercises for the reader to carry out. On completion, more points are introduced and so on. This makes the book fun to use because one can actually get on the computer while learning, rather than just reading the book by itself.

All the important niceties of programming that some programmers think they can avoid are introduced painlessly as a matter of course. I am talking in particular of remarks to indicate program title, version and author, **and the use of structure diagrams**. Extensive use is made of structure diagrams throughout the book. On completion of the course the student should realise just how valuable they are.

Another feature of this book that I think is unique is that some of the exercises are deliberately designed so the answers don't work! Just like real-life programming. A valuable way of learning.

In reading the text I have found a couple of errors. One is caused by a change to **SELECT OUTPUT "LP:"** in standard COMAL. The other relates to obtaining random integers, which will not apply to Commodore COMAL versions as they have a built in function to do this.

All in all, this a good teaching book that I think the newcomer to structured programming will find helpful and fun to read. ■

Control Key Values

By Larry Winckles

KEY / ASCII / FUNCTION

<Ctrl-@> 0 ???

<Ctrl-A> 1 Removes indentations of listed line, including wrap lines--Restores line to original state (before hitting <RETURN>)--Displays entire line after typing only the line number

<Ctrl-B> 2 Moves the cursor back one word (current line only)

<Ctrl-C> 3 Aborts program currently being run (Same as <STOP>)

<Ctrl-D> 4 Dumps graphics screen to printer

<Ctrl-E>* 5 Changes cursor color to white (Same as <Ctrl-2>)

<Ctrl-F> 6 Moves the cursor forward one word (current line only)

<Ctrl-G> 7 ???

<Ctrl-H>* 8 Disables upper/lower case toggle (<C= & Shift>)

<Ctrl-I>* 9 Enables upper/lower case toggle (<C= & Shift>)

<Ctrl-J> 10 ???

<Ctrl-K> 11 Clears text from present cursor position to end of current line

<Ctrl-L> 12 Moves cursor one space after last character on current line

KEY / ASCII / FUNCTION

<Ctrl-M>*13 Same as <RETURN>

<Ctrl-N>*14 Switches to lower case mode

<Ctrl-O> 15 ???

<Ctrl-P> 16 Sends text screen to printer

<Ctrl-Q>*17 Moves cursor down one line (Same as <CRSR-DOWN>)

<Ctrl-R>*18 Turns reverse mode on (Same as <Ctrl-9>)

<Ctrl-S>*19 Moves cursor to upper left hand corner of screen (Same as <HOME>)

<Ctrl-T>*20 Deletes character to the left of the cursor--In COMAL 2.0 only, when left border is encountered, deletes character immediately under the cursor (Same as)

<Ctrl-U> 21 Toggles between Graphics mode function key definitions and Edit mode function key definitions

<Ctrl-V> 22 Sets text background and border to blue, text cursor to white

<Ctrl-W> 23 Sets text background to light grey, text border to dark grey, cursor to black

<Ctrl-X> 24 Changes text border to color specified by the following <Ctrl-number> or <C=-number> color key

More ►

[illegible]

```

<Ctrl-Y> 25 Changes text background to
           color specified by the
           following <Ctrl-number> or
           <C=-number> color key

<Ctrl-Z> 26 Makes the current text
           background, border, and
           cursor colors the default

<Ctrl-:;> 27 ??? (Escape code?)

<Ctrl-\>*28 Changes cursor color to red
           (Same as <Ctrl-3>)

<Ctrl-;*>29 Moves cursor one position to
           the right (Same as
           <CRSR-RIGHT>)

<Ctrl- 30 Change cursor color to green
up-arrow> (Same as <Ctrl-6>)

<Ctrl-=*>31 Changes cursor color to blue
           (Same as <Ctrl-7>)

```

The following control codes should be familiar to all since they are the same as in BASIC 2.0 and COMAL 0.14:

```
<Ctrl-1> 144 Changes cursor color to
              black

<Ctrl-2>    5 Changes cursor color to
              white (Same as <Ctrl-E>)

<Ctrl-3>   28 Changes cursor color to red
              (Same as <Ctrl-\>)

<Ctrl-4>  159 Changes cursor color to
              cyan
```

```
<Ctrl-5> 156 Changes cursor color to
purple

<Ctrl-6> 30 Changes cursor color to
green (Same as <Ctrl-^>)

<Ctrl-7> 31 Changes cursor color to
blue (Same as <Ctrl-=>)

<Ctrl-8> 158 Changes cursor color to
yellow

<Ctrl-9> 18 Turns reverse mode on (Same
as <Ctrl-R>)

<Ctrl-0> 146 Turns reverse mode off
```

ASCII Conversion In COMAL 2.0, COMAL
TODAY #9, page 19
Displayed Codes In COMAL 2.0, COMAL
TODAY #9, page 20
Display Key Values, COMAL TODAY #9,
page 21
Function Keys, COMAL TODAY #8, page 33
Character Codes, COMAL TODAY #8, page 61
Character ROM, COMAL TODAY #6, page 37
COMAL 2.0 Auto ASCII Conversion, COMAL
TODAY #6, page 40
Case Lock & Unlock, COMAL TODAY #6,
page 41
Quote Mode Control Characters, COMAL
TODAY #6, page 49
Special CHR\$ Values, COMAL TODAY #4,
page 37
Cartridge Graphics and Sound, page 63
Cartridge Tutorial Binder, Appendix D,
page 285
Commodore 64 Programmers Reference
Guide, page 93
Commodore 64 Programmers Reference
Guide, Appendix C, page 379
Commodore 64 Users Guide, Appendix F,
page 135

Fix Disk Errors

by David Stidolph

It's late at night, you've been working on your "ultimate" data base program till you get it just right. The power goes off, but your not worried, you've been making backups of your program every 15 minutes like you're supposed to. This time, however, when you go to reload your last version, the program never loads because the disk drive has reported a READ ERROR. Don't panic (yet), just run the following program (the program is also on the 2.0 side of *Today Disk #11*). If the error is one of the two most common on Commodore disk drives (22, checksum error of the data block, or 23, data block ID error) this program can fix it.

The program first checks every sector on each track and makes a note of any errors. Once the entire disk has been checked, it displays a map of all errors and begins to correct any bad, but repairable sectors.

Disk errors can be broken down into two groups, repairable and non-repairable. Repairable errors are those which involve corrupted data, and not alignment or "hardware" problems. Corrupted data in a sector means either the checksum is bad (which is fine) or the information has been changed (this is not so fine). It is even possible that the first two bytes which point to the next track and sector are bad (this is a disaster). This program can correct the sector so that it will no longer report an error, but the information in the sector may be corrupted from what it was supposed to be originally. If the track and sector numbers were altered, you will have to use a "disk editor" to correct them.

```
// delete "correct'disk"
// save "correct'disk"
// by David Stidolph
//
USE system
textcolors(0,0,7)
PAGE
PRINT CHR$(14)
PRINT "This program will";
PRINT "cycle through";
PRINT "all sectors on each";
PRINT "track, check";
PRINT "for errors and attempt";
PRINT "to recover";
PRINT "bad sectors."
PRINT
PRINT "Please insert disk and";
INPUT "press RETURN ": dummy$
PRINT
PRINT "This test takes 2-3 minutes!"
PRINT "Please wait."
PRINT
TRAP
MOUNT
HANDLER
e:=recover'block(18,0)
IF e<>1 THEN
    PRINT "Cannot work on this disk!!"
    STOP ERRTEXT$
ENDIF
ENDTRAP
DIM error(35,0:21)
disk'errors(error(,)); show'disk
FOR trk#:=1 TO 35 DO
    FOR sec#:=0 TO max'sector(trk#) DO
        IF error(trk#,sec#)=4 OR error(trk#,sec#)=5 THEN
            textcolors(-1,-1,13)
            PRINT AT 25,1: "Attempting to";
            PRINT "recover TRACK";trk#;
            PRINT "Sector";sec#;
            error(trk#,sec#):=recover'block(trk#,sec#)
            show'disk
        ENDIF
    ENDFOR sec#
ENDFOR trk#
textcolors(-1,-1,13)
PRINT AT 25,1: "Test done --";
PRINT "Press RETURN to continue";
WHILE KEY$<>"13" DO NULL
PAGE
END "End of test!"
//
FUNC recover'block(trk,sec) CLOSED
IMPORT seek'block,read'block
IMPORT write'block,job,max'sector
DIM drive$ OF 4
e:=0
IF trk>0 AND trk<36 THEN
    max:=max'sector(trk)
    IF sec>=0 AND sec<=max THEN
        drive$:=drive'type$; buf:=0
        e:=seek'block(trk,sec,buf)
        IF e=1 THEN
```

More ►

Fix Disk Errors - continued

```

e:=read'block(trk,sec,buf)
IF e=4 OR e=5 THEN
  e:=write'block(trk,sec,buf)
ENDIF
ENDIF
ENDIF
RETURN e
ENDFUNC recover'block
//
FUNC max'sector(trk) CLOSED
IF trk<18 THEN
  RETURN 20
ELIF trk<25 THEN
  RETURN 18
ELIF trk<31 THEN
  RETURN 17
ELSE
  RETURN 16
ENDIF
RETURN max
ENDFUNC max'sector
//
FUNC seek'block(t,s,b)
  RETURN job($b0,t,s,b)
ENDFUNC seek'block
//
FUNC read'block(t,s,b)
  RETURN job($80,t,s,b)
ENDFUNC read'block
//
FUNC write'block(t,s,b)
  RETURN job($90,t,s,b)
ENDFUNC write'block
//
FUNC job(n,t,s,b) CLOSED
  IMPORT disk'write,disk'read
  try:=0
  disk'write(b*2+6,t)
  disk'write(b*2+7,s)
  disk'write(b,n)
  REPEAT
    try:=+1
    c:=disk'read(b)
  UNTIL c<128 OR try>500
  RETURN c
ENDFUNC job
//
PROC disk'errors(REF error(,)) CLOSED
  IMPORT max'sector
  OPEN FILE 2,"u8:#2/s2"
  FOR track#:=1 TO 35 DO
    max#:=max'sector(track#)
    count#:=0; sector#:=0
    REPEAT
      count#:=+1
    TRAP
      PASS "u1: 2 0 "+STR$(track#)+" "+STR$(sector#)
      error(track#,sector#):=1
    HANDLER
      error(track#,sector#):=ERR-218
    ENDTRAP
    sector#:=+2
    IF sector#>max# THEN sector#:=1
    UNTIL count#>max#
  ENDFOR track#
  CLOSE FILE 2
ENDPROC disk'errors
//
PROC show'disk
  PAGE
  USE system
  textcolors(-1,-1,13)
  PRINT "  TRACK";
  textcolors(-1,-1,14)
  PRINT "1111111111112222222222333333"
  PRINT " ",
  FOR x:=1 TO 35 DO PRINT x MOD 10,
  PRINT
  FOR sector:=0 TO 20 DO
    IF sector<6 THEN
      textcolors(-1,-1,13)
      PRINT "SECTOR"(sector+1),
      textcolors(-1,-1,14)
      PRINT sector;
    ELSE
      textcolors(-1,-1,14)
      PRINT USING "##": sector;
    ENDIF
  FOR track:=1 TO 35 DO
    CASE error(track,sector) OF
      WHEN 1
        textcolors(-1,-1,1)
        PRINT CHR$(186),
      WHEN 4,5
        textcolors(-1,-1,7)
        PRINT "+",
      WHEN 0
        PRINT " ",
      OTHERWISE
        textcolors(-1,-1,2)
        PRINT "-",
    ENDCASE
  ENDFOR track
  PRINT
  ENDFOR sector
  PRINT ""154""",CHR$(91),"5""",CHR$(186),
  PRINT ""154"] OK ["158"+"154"]";
  PRINT "REPAIRABLE ["28"-154"] BAD SECTOR"
ENDPROC show'disk
//
FUNC disk'read(addr) CLOSED
  DIM com$ OF 20
  com$:= "m-r"+CHR$(addr MOD 256)
  com$:=+CHR$(addr DIV 256)+"1"
  PASS com$
  RETURN ORD(STATUS$)
ENDFUNC disk'read
//
PROC disk'write(addr,num) CLOSED
  DIM com$ OF 20
  com$:= "m-w"+CHR$(addr MOD 256)
  com$:=+CHR$(addr DIV 256)+"1"
  com$:=+CHR$(num)
  PASS com$
ENDPROC disk'write

```

Your RUN / POP Key

by Len Lindsay

Yuk! Awful screen colors! I wish I could change them. Now you can. Just use your POP key.

Hey, what's on that disk? But I don't want to stop the program. No problem. Use your POP key.

But where is the POP key? Ah, your POP key must have been mislabeled too. Our key top says RUN/STOP instead of RUN/POP. OK. So that's your POP key. How it works is the point of this article.

First some background. COMAL 2.0 for the CBM 8032 computer included a new keyword, INTERRUPT, to monitor SRQ line on the IEEE-488 bus. Few of us had IEEE devices that provided your program with SRQ interrupts so this command was ignored. Even so, the INTERRUPT keyword was included in the C64 COMAL 2.0 cartridge and remained virtually unused. On July 17, 1985 we published *Using the Interrupt Command* by Jesse Knight, on page 62 of *COMAL Today* #8. Jesse provided a very small *SETUP* routine that caused the *STOP* key to create an interrupt. Surprisingly enough, it remained virtually unused even after this article was published.

Two other commands have been generally ignored: GETSCREEN and SETSCREEN, part of the SYSTEM package. We included a demo program on *Cartridge Demo Disk* #2 called "*get/set'screen*" illustrating their use. They came in handy while creating a HELP screen for *Rod the Roadman* on *Today Disk* #9.

Meanwhile, IBM PC users enjoy (and pay \$50 for) a cute utility program called *SideKick*, which can pop up a menu window

and provide the user with a calculator, notepad, and calendar. The user pops it up, uses it briefly, and hits the ESC key. His screen then returns back to its original state. Users marvelled at this program. It became a best seller and was named product of the year.

Now, as you might have already guessed, our neglected COMAL 2.0 commands will come together to provide you with a pop over system, just like the IBM PC users have (and pay \$50 extra for). Only ours can be customized to fit exactly what you want. Yes, it is written entirely in COMAL (except for the *SETUP* routine) and is small enough to be MERGED onto any of your programs. Or, you can incorporate the POPOVER system directly into your program as an integrated menu system.

To explain how the POPOVER system works, we will create one now, step by step. First, the STOP key must be redefined so that it will cause an interrupt when pressed. The *SETUP* procedure does this. Next, we want to save the starting text screen so that when we are done we can replace it. One line does this:

```
getscreen(start'screen$)
```

Note that start'screen\$ is previously DIMmed for 1505 characters. Now, we can pop up a menu and do whatever we want. When done, the following line returns the screen as it was when we started:

```
setscreen(start'screen$)
```

The POPOVER system is smart. No one will see the menu if the graphics screen is active. So, if the graphics screen is currently displayed, it flips in the text screen. When done it returns the text screen back to its original state

More ►

and redisplay the graphics screen if it was originally active. The POPOVER system also clears the keyboard buffer as it starts and exits, and resets the status of the STOP key. It also turns off the INTERRUPT when it starts to avoid recursive POPS. It is turned on again at the end.

The POPOVER system is entirely self contained. Look closely at the system. Notice that it is one big procedure called POPOVER. But inside POPOVER are other procedures, such as SETUP. The popover system could also have been constructed from many individual procedures. IMPORT would have worked just as well. But the advantage of nested procedures is that it provides everything in one big chunk. To see the entire system we can type:

LIST popover

You may have guessed already that this makes it easy to create a ready to merge **POPOVER** file on disk:

LIST popover "pop.test"

Use the MERGE command to add POPOVER to your programs. No need to individually MERGE each procedure in the system. And the system can be activated from any program by adding only one line at the very beginning of the program:

0001 popover

Yes, that is all you have to add to your program. Almost. The exception (there always are exceptions) is CLOSED procedures. Inside each CLOSED procedure in your program you must add this line:

IMPORT popover

As you create different POPOVER systems, start the filenames with "POP." for easy identification. Programs that have a POPOVER attached can include ".POP" at the end of the filename. These POPOVER systems are on *Today Disk #11*:

POP.SKELETON

POP.DIR

POP.COLORS

POP.DIR+COLORS

The **DIR** system allows you to POP a DIRectory while a program is running (be careful if there are files open!).

The **COLORS** system allows you to change the color settings of the text screen. Remember that **SETSCREEN** returns the original colors as well as the text. To get around this, change the first three bytes of **start'screen\$** (which are the color bytes).

DIR+COLORS, is simply a combination of both the DIR and the COLOR system.

Finally, the SKELETON system (listed below) is a complete POPOVER without any extra options. Just Q to Quit and *<return>* key to return. Use it to start building your own system. If everyone uses the same SKELETON for their POPOVER system, they will be compatible. The Q key will always mean Quit (not E for End, X to eXit, etc). Plus, if a user accidentally hits STOP, the *<return>* key will escape from the POPOVER system back to the program.

An example program is also listed after this article. It includes the **COLORS** system. Type it in. Then issue these commands:

More ►

Your RUN / POP Key - continued

SAVE "example.pop"
LIST popover "pop.colors"

You now have a MERGEable POPOVER system on your disk, as well as the example program. Now, RUN the program to see how easy it works. The example program prints the current status of the 34 different INQ registers. If you change the colors (hit the STOP, I mean POP key) watch the values of 1, 2, and 3. They keep the text screen color values.

Final note: as it stands, the POP key will not work during an INPUT request. Can some COMALite please tell us how to POP during INPUT? Also, the POP menu always shows up when the program starts. Is that OK? Finally, yes, it works fine with user defined fonts!

Chart of current INQ(x) values

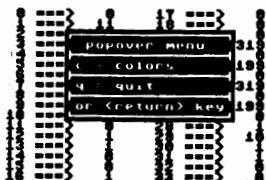


Chart of current INQ(x) values



POP.SKELETON

```
PROC popover CLOSED
  INTERRUPT //interrupt off while doing interrupt
  USE graphics
  textmode:=inq(13)
  graphmode:=inq(7) //2=multi // 1=full hi-res
  IF graphmode<2 THEN graphmode:=1-inq(14)
  TRAP ESC- // 0=splitscreen
  setup
  USE system
  DIM start'screen$ OF 1505
  getscreen(start'screen$)
  IF NOT textmode THEN textscreen
  clear'keys
  popup
  setscreen(start'screen$)
  IF textmode THEN
    textscreen
  ELSE
    IF graphmode THEN
      fullscreen
    ELSE
```

```
      splitscreen
    ENDIF
  ENDIF
  clear'keys // optional line
  INTERRUPT popover
  //
  PROC popup
    col:=RND(3,15) // <<<===start col
    current'row:=RND(2,12) // <<<=== start row
    // row is a function that starts at current'row
    // use shift * to draw menu line
    PRINT AT row,col: ""18"+-----+"
    PRINT AT row,col: ""18" popover menu "
    PRINT AT row,col: ""18"+-----+"
    PRINT AT row,col: ""18" = "
    PRINT AT row,col: ""18" = "
    PRINT AT row,col: ""18"+-----+"
    PRINT AT row,col: ""18" q = quit "
    PRINT AT row,col: ""18"+-----+"
    PRINT AT row,col: ""18" or <return> key "
    PRINT AT row,col: ""18"+-----+"
    REPEAT
      done'popping:=TRUE
      CASE KEY$ OF
        WHEN "q","Q"
          TRAP ESC+
          END ""147"Thank You."
        WHEN ""13"" //carriage return
          RETURN
        OTHERWISE
          done'popping:=FALSE
      ENDCASE
    UNTIL done'popping
  ENDPROC popup
  //
  PROC ready
    INPUT AT 25,1,0: ""18"Press RETURN when ready": rd$
    PAGE
  ENDPROC ready
  //
  FUNC row
    current'row:=+1
    RETURN current'row
  ENDFUNC row
  //
  PROC clear'keys
    WHILE KEY$>"" DO NULL
      dummysc:=ESC //clear stop key
    ENDPROC clear'keys
  //
  PROC setup CLOSED
    TRAP ESC- // setup by jesse knight
    FOR x#:=0 TO 12 DO
      READ byte#
      POKE $c86a+x#,byte#
    ENDFOR x#
    POKE $c7e2,$6a
    POKE $c7e3,$c8
    POKE $4d,PEEK($4d) BITOR $20
    DATA $a5,$4d,$29,$08,$f0,$06,$a9
    DATA $04,$05,$4d,$85,$4d,$60
  ENDPROC setup
ENDPROC popover
```

More ►

```

start'screen$(3):=CHR$(inq(3))
WHEN "q","Q"
  TRAP ESC+
  END ""147"Thank You."
WHEN ""13"" //carriage return
  RETURN
OTHERWISE
  done'popping:=FALSE
ENDCASE
UNTIL done'popping
ENDPROC popup
//
PROC ready
  INPUT AT 25,1,0: ""18"Press RETURN when ready": popready$
  PAGE
ENDPROC ready
//
FUNC row
  current'row:+1
  RETURN current'row
ENDFUNC row
//
PROC clear'keys
  WHILE KEY$>" DO NULL
    dummyesc:=ESC //clear stop key
  ENDPROC clear'keys
//
PROC setup CLOSED
  TRAP ESC- // setup by jesse knight
  FOR x#:=0 TO 12 DO
    READ byte#
    POKE $c8a+x#,byte#
  ENDFOR x#
  POKE $c7e2,$6a
  POKE $c7e3,$c8
  POKE $4d,PEEK($4d) BITOR $20
  DATA $a5,$4d,$29,$08,$f0,$06,$a9
  DATA $04,$05,$4d,$85,$4d,$60
ENDPROC setup
//
PROC set'colors CLOSED
  USE system
  USE graphics
  PAGE
  LOOP
    PRINT AT 3,1: ""18"set colors now"
    PRINT AT 6,1: "press "18" f1 "146" border color"
    PRINT AT 8,1: "press "18" f3 "146" background color"
    PRINT AT 10,1: "press "18" f5 "146" text color"
    PRINT AT 13,1: "press "18" f7 "146" or "18" q "146" quit"
    CASE KEY$ OF
      WHEN ""133""
        textcolors((inq(1)+1) MOD 16,-1,-1)
      WHEN ""134""
        textcolors(-1,(inq(2)+1) MOD 16,-1)
      WHEN ""135""
        textcolors(-1,-1,(inq(3)+1) MOD 16)
      WHEN ""136"","q","Q"
        EXIT
    OTHERWISE
      ENDCASE
  ENDOLOOP
ENDPROC set'colors
ENDPROC popover

```

2.0 Disk Directory Tips

Both the CAT and DIR commands allow you to see a directory of the files on your disk in the disk drive (we will use DIR in this article). Type:

DIR

To pause the directory just press the SPACE bar. Press the SPACE bar again and the directory resumes. If you see a program you wish to RUN, hit the STOP key. Your cursor appears on the next line. Move the cursor up to the line with the program you want to RUN. Now, hit the F7 key and the program is first LOADED and then automatically RUN.

Now, for some extra capabilities. If you want to just LOAD the program, but not RUN it, type in the word LOAD instead of pressing the F7 key. Then hit the <RETURN> key. You will get an error and the cursor moves on top of the letter P in PRG. Now just press the SPACE bar 3 times and hit <RETURN> again and the program will load (CONTROL K may be used instead of the 3 spaces).

If your directory is rather long, you can use pattern matching to have just selected files displayed. Here are some examples with explanations:

DIR "2:"

This is how to see the directory of a drive that is not the current drive. In this case, it was drive "2:" which is drive 0 of device 9.

DIR "TEST"

This will list the disk header info, only the files whose names match "TEST" and the number of blocks free. Normally only one file can have the name TEST so

only it could match. This is a quick way to make sure a file is on a disk, or to see how many blocks are free without displaying all the files.

DIR "A*"

This will display only the files whose names start with A. The "*" means ignore the rest of the file name while checking for a match. If it matches up to the * then it is considered a match.

DIR "????"

A "?" means any character can match in this position. This example would list all files that had exactly 4 characters in the filename. "TEST" and "AB33" would match. "T2" and "TESTING" would not.

DIR "SHAP.*"

This lists all files whose names start with "SHAP.". This is useful if you prefix the names of your files as recommended (see *Filename Conventions*).

DIR "*s"

The "*" means all the files. The "s" means SEQ type only. Thus all the SEQ type files on the disk are displayed.

DIR "*p"

Same as the previous example, except only PRG type files are displayed.

You can change the current drive to be "2:" if you are using device 9 drive 0. This command will do it:

UNIT="2:" ■

Take Off With Us

◆ ICCE's the one for you ◆

With today's profusion of computer information, it's hard to know where to start—which road to choose. The International Council for Computers in Education has been guiding the way in the computer education field since 1979, providing leadership and a ground plan for the future.

It's the one organization every computer educator, administrator, coordinator or librarian needs.

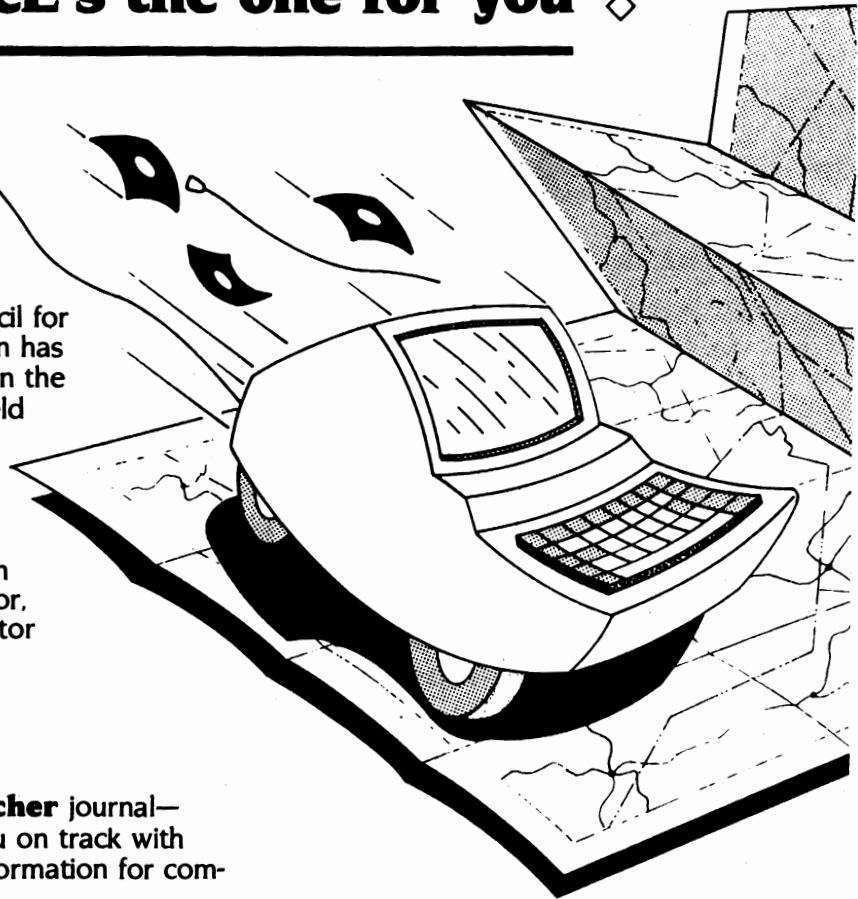
It's the one for you.

GUIDING THE WAY

The Computing Teacher journal—guaranteed to keep you on track with up-to-date, practical information for computers in the classroom.

Special Interest Groups—share information to help your special interest area grow. SIGs include computer coordinators, teacher educators, administrators and special educators, and are planned for advanced placement in computer science, community colleges and videodisc users. The quarterly **SIG Bulletin** serves as a forum for SIG information.

Booklets and Monographs point to additional information on specific topics. **ICCE Packets** provide you with teacher training materials. Members receive a 10% discount on all three.



ICCE Committees address a variety of ethical and practical issues important to you as a computer-using educator.

ICCE participates in computer education conferences throughout the world, supporting our state- and region-wide member organizations.



Join the One for You.

COMAL Clinic

FUNCTION KEY LINE ENTRY

In COMAL 2.0 the function keys can be defined to assist you in entering a program. For instance, if you will be entering a lot of data statements you may wish to redefine the *F1* key. Type:

```
USE system
defkey(1,""13"DATA ")
AUTO
```

Press *F1*. Enter a data line as usual but use the *F1* key instead of the *<RETURN>* key at the end of each line. Notice that it does a return to the next line and also prints the word DATA for you!

SUBSTRING SHORTCUT

David Stidolph has discovered that a substring shortcut documented in IBM PC COMAL will also work on the C64 COMAL 2.0 Cartridge. The proper way to specify a substring is to include the starting character and the ending character inside parentheses like:

```
TEXT$(4:6)
```

That specifies the 4th thru 6th characters of TEXT\$. Now, if you want all the characters from 4 through the end of the string, you can use this:

```
TEXT$(4:)
```

Just leave off the ending character number after the colon (:) and COMAL assumes you mean the last character. If you leave off the starting number before the colon, COMAL assumes you mean the first character:

```
TEXT$( :6)
```

RECOVER A LOST FILE

Have you ever DELETED the wrong program on a disk? Well, you can recover the program (or data file) quite easily. Let's say that I am writing a program I call TEST5 and I go to delete the old version, but type DELETE "TESTS" which is a very important program I worked all night on. I can recover TESTS by just doing a LOAD "*". The disk drive interprets an asterisk as the last file accessed and doesn't care if has been DELETED. Once loaded it can be SAVED again. Of course this will destroy whatever is in memory, but it does give you the ability to restore what was lost.

BACK TO COMAL

Ian MacPhedran provides this tip. If you often use a program which operates in the BASIC environment, and which wipes out the COMAL reboot routine (sys 50000), you don't have to reset the machine to get back in COMAL. Simply SAVE this routine onto the disk once by the following procedure:

Go to BASIC from COMAL 2.0. In BASIC:

```
poke 43,80
poke 44,195
poke 45,112
poke 46,195
save "comal'boot",8,0
poke 43,1
poke 44,8
new
```

After this, you can reboot COMAL 2.0 by:

```
load "comal'boot",8,1
sys 50000
```

More ►

FILE TIPS

It is very important to always close disk files. If you neglect to close a WRITE or APPEND file, it becomes what is called a *splat* file. If it is not closed, the Disk Operating System is upset. Since there is no end to the file, the DOS will normally not let you OPEN it. This is very frustrating. How can you fix this messed up file if you can't even access it anymore? There is a way! When a file is opened as a READ file, COMAL sends the code ("r") with the filename to the disk drive. COMAL also sends ("w") for a WRITE file and ("a") for an APPEND file. COMAL will allow you to override this code that must be sent to the disk drive. For instance:

OPEN FILE 2,"0:name,p,r"

The ",p" is short for ",prg" which means it is a PRG type file. The ",r" is short for ",read" which means to open the file as a READ file. This line will open a program file as if it were a data file.

All you need to know now is that the DOS also accepts ",m" as a wildcard that matches anything - even a *splat* file. So, to read your file use:

OPEN FILE 2,"0:name,s,m"

This will open a sequential *splat* file.
The next line will open a PRG type *splat* file:

OPEN FILE 2,"0:name,p,m"

While reading the file, be prepared for strange things to happen when it gets to the point where there is "no end".

CAN YOU FIGURE THIS OUT?

Many people have told us that playing with the keyboard buffer from COMAL 2.0 did not work any more (it worked in COMAL 0.14 just as it did in BASIC). Well, then can you explain this test? Just enter the procedure below, then issue the command: SCAN. Next type: TEST. When you hit return, the screen will clear, the word CAT prints, then a catalog of your disk appears. Hmmm... The dynamic keyboard seems to work. Now, remove the // in front of the END statement. Now try it again. Hmmm... Now it doesn't work. Can you solve this puzzle? Let us know if you do.

```
0010 PROC test
0020   USE system
0030   POKE 198,1 //buffer count
0040   POKE 631,13 //key value
0050   PAGE
0060   PRINT AT 2,1:"cat"
0070   CURSOR 2,1
0080   //END ""
0090 ENDPROC test
```

CLEAR INPUT AT FIELD

COMAL 2.0 provides you with protected input fields, and INPUT AT allows you to put a limit on the field length. However, sometimes you may wish to clear the field at the beginning automatically. Well, Joel Rea has informed us that the following lines will clear the INPUT AT field if issued just prior to the INPUT AT command:

```
POKE $C866,$B1
POKE $C867,$C7
POKE $C7B1,147
POKE $C865,1
```

C128 80 Column Screen

by David Stidolph

Commodore said that their new Commodore 128 computer system would be 100% compatible with existing C64 software, and would "lock out" the extra features of the C-128 while in "64 mode". Since the COMAL cartridge only runs in the C64 mode of the C128, none of these new features (such as 80 column) were *supposed* to be accessible. From BASIC, sure. But not from COMAL! I found that the 80 column video chip IS accessible from 64 mode. Printing information on the 80 column screen is difficult. However, if you use the procedures listed following this article, most of the work will be done for you.

SAMPLE COMAL PROGRAM

An example COMAL 2.0 program is listed following this article. It initializes the VDC and writes the character definitions into its RAM (this takes a couple minutes). Then it clears the screen and displays some text. Flash, underline, and color changes are all shown. The program will also turn on the cursor so you can see it and allow you to type, change colors with the color keys, and try out underline, reverse, and blinking text. Note that the cursor does not show what color is selected, only what color the "current" attribute ram location is set at.

IMPORTANT ROUTINES

FLASH(state)
REVERSE(state)
UNDERLINE(state)
ALTERNATE(state)

These procedures "activate" or "deactivate" the specified attribute. As

characters are displayed, the corresponding active attributes are also set for that character. Memory location 2024 is used to store the current copy of the attributes for writing to the VDC. For example, FLASH(TRUE) would set flashing mode on. All characters printed from then on would be flashing (until a FLASH(FALSE) was issued). ALTERNATE specified which of the two character sets to use (thus characters from both can be on the screen at the same time).

PRINT80(row,col,text\$)

This procedure displays the text contained in the third parameter (text\$) on the 80 column screen beginning at the location specified by the first parameter (row) and the second parameter (col). Any active attributes (such as flashing or underline) are also set for the characters. The top left-hand corner of the screen is 1,1. The bottom right-hand corner of the screen is 25,80. Here are some examples:

```
print80(1,1,"Cursor HOME position")
text$="This is a sample string"
print80(12,15,text$)
```

CLEAR80

This procedure blanks the 80 column text screen and turns off all attributes in the video RAM. Active attributes remain unchanged at memory location 2024.

PAGE80

Deactivates all attributes and executes CLEAR80.

SCROLL

This procedure scrolls the screen, and

More ►

its attributes, up one line and blanks the bottom line.

MOVECURSOR(row,col)

This procedure moves the hardware cursor to the screen position specified. The following example moves the cursor to the HOME position:

```
movecursor(1,1)
```

SETCURSOR(first,last,mode)

This procedure sets the cursor attributes. The first parameter sets what raster line the cursor begins on. The second parameter sets the ending raster line. The defaults are 0 and 7 (a block cursor). The last parameter sets the mode, or state of the cursor. The following table shows what the mode can be set to:

```
mode:=0 // Non-blinking  
mode:=1 // Cursor not displayed  
mode:=2 // Blink fast  
mode:=3 // Blink normal speed
```

INITCHARS(bank,vdcset)

This procedure writes the font set defined in the first parameter (0-3) into the VDC character set area (0-1). This takes a couple minutes.

TYPE(row,col)

This procedure emulates the keyboard input and screen output routines of the C128. The two parameters specify where to put the cursor and begin the text display.

```
// delete "c64mode80col"
// save "c64mode80col"
// by David Stidolph
//
foreground:=1; back:=0
PAGE
PRINT "Initializing 80 column screen"
PRINT
PRINT "Please come back in a couple of"
PRINT "minutes and switch to 80 column screen"
PRINT "via the RGBI cable on the back"
PRINT "of the C128."
PRINT
PRINT "The 40/80 column switch on the keyboard"
PRINT "is not used by this program."
init80
initchars(3,0)
//
color80(2,TRUE)
page80
//
flash(TRUE)
underline(TRUE)
print80(1,27,"This is the 80 column screen!")
flash(FALSE)
underline(FALSE)
//
color80(3,FALSE)
row:=2
DIM text$ OF 80
WHILE NOT EOD DO
  READ text1$,text2$
  text$:=text1$+text2$
  row:=+1
  print80(row,1,text$)
ENDWHILE
color80(7,TRUE)
print80(row,15," ")
movecursor(row,15)
setcursor(0,7,3)
type(row,15)
//
DATA "This text is being printed by the COMAL"
DATA " cartridge and it does it very well."
DATA "Soon we may have a very fast machine"
DATA " language package to do this"
DATA " and will make output to this"
DATA " type of screen easy."
DATA "" "" // blank line
DATA "80 column output will not affect"
DATA " the normal 64 either!"
DATA "(of course nothing would be printed"
DATA " on the screen)"
DATA "" ""
DATA "Feel free to type on this screen --"
DATA " Most keys are active including the"
DATA "color control keys and the function keys"
DATA "."
DATA "" ""
DATA "F1) Reverse On      "
DATA "F2) Reverse Off     "
DATA "F3) Underline On    "
DATA "F4) Underline Off   "
```

More ►


```

DATA "F5) Flash On" "
DATA "F6) Flash Off" "
DATA "F7) Increment Background color",""
DATA "" ""
DATA "The cursor ","=>"
//
PROC set(reg,num)
    POKE $d600,reg
    POKE $d601,num
ENDPROC set
//
PROC setw(reg,addr)
    set(reg,addr DIV 256)
    set(reg+1,addr MOD 256)
ENDPROC setw
//
PROC init80 CLOSED
    IMPORT set
    count:=0
    WHILE NOT EOD DO
        READ num
        set(count,num)
        count:=+1
    ENDWHILE
    POKE 2024,%00001111
    //
    DATA 126,80,102,73,39,224,25
    DATA 32,252,231,160,231,0,0
    DATA 0,0,0,0,0,16,0,120,232,32
    DATA 64,240,0,47,231,0,0,0,0,125
    DATA 102,245
ENDPROC init80
//
PROC print80(row,col,text$) CLOSED
    IMPORT set,setw
    row:=-1; col:-1
    address:=row*80+col
    color:=address+4096
    num:=LEN(text$)
    FOR x:=1 TO num DO
        char:=ORD(text$(x))
        IF char>63 THEN char:-64
        IF char>128 THEN char:-64
        print'at(address+x-1,char)
        print'at(color+x-1,PEEK(2024))
    ENDFOR x
    //
    PROC print'at(addr,c)
        setw(18,addr)
        set(31,c)
    ENDPROC print'at
ENDPROC print80
//
PROC movecursor(row,col) CLOSED
    IMPORT set,setw
    row:=-1; col:-1
    address:=row*80+col
    setw(14,address)
ENDPROC movecursor
//
PROC setcursor(first,last,mode) CLOSED
    // mode=0 => non-blinking
    // mode=1 => cursor not displayed

```

```
// mode=2 ==> blink fast
// mode=3 ==> normal blink
// first=start raster line for cursor
// last=last raster line for cursor
IMPORT set,setw
num:=mode*32+first
set(11,last)
set(10,num)
ENDPROC setcursor
//
PROC scroll
set(24,128)
FOR row:=0 TO 21 STEP 3 DO
    setw(18,(row*80))
    setw(32,((row+1)*80))
    set(30,240)
    setw(18,(row*80)+4096)
    setw(32,((row+1)*80)+4096)
    set(30,240)
ENDFOR row
set(24,0)
setw(18,24*80)
set(31,32)
set(30,79)
setw(18,24*80+4096)
set(31,PEEK(2024))
set(30,79)
ENDPROC scroll
//
PROC clear80
FOR pag:=0 TO 7 DO
    setw(18,pag*256)
    set(31,32)
    set(30,0)
ENDFOR pag
color:=PEEK(2024) MOD 16
FOR pag:=16 TO 23 DO
    setw(18,pag*256)
    set(31,color)
    set(30,0)
ENDFOR pag
ENDPROC clear80
//
PROC page80
flash(FALSE); underline(FALSE)
alternate(FALSE); reverse(FALSE)
clear80
ENDPROC page80
//
FUNC attributes
RETURN PEEK(2024)
ENDFUNC attributes
//
PROC flash(state) CLOSED
old:=PEEK(2024) BITAND %11101111
POKE 2024,old BITOR (state>0)*16
ENDPROC flash
//
PROC underline(state) CLOSED
old:=PEEK(2024) BITAND %11011111
POKE 2024,old BITOR (state>0)*32
ENDPROC underline
//
```

More ►

```

PROC reverse(state) CLOSED
old:=PEEK(2024) BITAND %10111111
POKE 2024,old BITOR (state>0)*64
ENDPROC reverse
//
PROC alternate(state) CLOSED
old:=PEEK(2024) BITAND %01111111
POKE 2024,old BITOR (state>0)*128
ENDPROC alternate
//
PROC color80(color,intensity) CLOSED
clr:=(color MOD 8)*2+(intensity<>0)
old:=PEEK(2024) BITAND %11110000
POKE 2024,old+clr
ENDPROC color80
//
PROC initchars(bank,vdcset) CLOSED
IMPORT set,setw
FOR pag:=32 TO 63 DO
    setw(18,pag*256)
    set(31,0)
    set(30,0)
ENDFOR pag
USE font
DIM c$ OF 8
FOR char:=0 TO 255 DO
    getcharacter(bank,char,c$)
    address:=8192+char*16+vdcset*4096
    FOR raster:=0 TO 7 DO
        setw(18,address+raster)
        set(31,ORD(c$(raster+1)))
    ENDFOR raster
    setw(18,address+8)
    set(31,0)
    set(30,7)
ENDFOR char
ENDPROC initchars
//
PROC type(row,col)
DIM c$ OF 1
LOOP
    c$:=KEY$
    CASE c$ OF
        WHEN ""13"" // RETURN
            row:=+1; col:=1
        WHEN ""19"" // HOME
            row:=1; col:=1
        WHEN ""147"" // CLR/HOME
            row:=1; col:=1
        page80
        WHEN ""145"" // CRSR up
            row:=-1
        WHEN ""17"" // CRSR down
            row:=+1
        WHEN ""29"" // CRSR right
            col:=+1
        WHEN ""157"" // CRSR left
            col:=-1
        WHEN ""18""""133"" // REVERSE on
            reverse(TRUE)
        WHEN ""146""""137"" // REVERSE off
            reverse(FALSE)
        WHEN ""20""""148"" // del&inst

```

```

NULL // Sorry! Not implemented
WHEN ""134"" // underline on
underline(TRUE)
WHEN ""138"" // underline off
underline(FALSE)
WHEN ""135"" // flash on
flash(TRUE)
WHEN ""139"" // flash off
flash(FALSE)
WHEN ""136"" // increment background
back:+1; back:=back MOD 16
set(26,foreground*16+back)
WHEN ""140"" // increment foreground
foreground:+1; foreground:=foreground MOD 16
set(26,foreground*16+back)
WHEN ""5""
color80(7,TRUE)
WHEN ""28"" // RED
color80(4,FALSE)
WHEN ""30"" // Green
color80(2,FALSE)
WHEN ""31"" // Blue
color80(1,FALSE)
WHEN ""144"" // Black
color80(0,FALSE)
WHEN ""156"" // Purple
color80(5,FALSE)
WHEN ""158"" // Yellow
color80(6,TRUE)
WHEN ""159"" // Cyan
color80(3,FALSE)
WHEN ""129"" // Orange
color80(3,TRUE)
WHEN ""149"" // Brown
color80(6,FALSE)
WHEN ""150"" // Light Red
color80(4,TRUE)
WHEN ""151"" // Dark Grey
color80(0,TRUE)
WHEN ""152"" // Medium Grey
color80(5,TRUE)
WHEN ""153"" // Light Green
color80(2,TRUE)
WHEN ""154"" // Light Blue
color80(1,TRUE)
WHEN ""155"" // Light Grey
color80(0,TRUE)
WHEN ""","0"" // nothing
NULL
OTHERWISE
print80(row,col,c$); col:+1
ENDCASE
IF col<1 THEN col:=80; row:-1
IF col>80 THEN row:+1; col:=1
IF row<1 THEN row:=1
IF row>25 THEN scroll; row:=25
movecursor(row,col)
ENDLOOP
ENDPROC type

```

COMAL 2.0 Keywords

compiled by Daniel W Parish
(++ after keyword means 0.14 too)

```
// ++ allows comments in a program
//[<anything>]
// next line scans name field
```

ABS ++ gives the absolute value
ABS(<numeric expression>)
PRINT ABS(standard-number)

AND ++ logical AND
<expression> AND <expression>
IF number>0 AND number<100 THEN

AND THEN -- logical AND extension
<expression> AND THEN <expression>
IF reply\$>" AND THEN "." IN reply\$ THEN

APPEND ++ start at end of seq file
OPEN [FILE] <file#>,<filename>,APPEND
OPEN FILE 2,"test.dat",APPEND

AT -- begin at specified location
PRINT AT<row>,<col>:[<prnt lst>][<mark>]
PRINT AT 1,1:"Section number:";num;

ATN ++ arctangent in radians
ATN(<numeric expression>)
PRINT ATN(num1+num2)

AUTO ++ automatic line numbering
AUTO [<start line>],[<increment>]
AUTO 9000

BASIC ++ back into BASIC mode
BASIC
BASIC

BITAND -- bitwise AND
<argument> BITAND <argument>
show(bnum BITAND %00001000)

BITOR -- bitwise OR
<argument> BITOR <argument>
PRINT (bnum BITOR flag)

BITXOR -- bitwise XOR
<argument> BITXOR <argument>
bnum=(num1+num2) BITXOR %10000000

CASE ++ multiple choice decisions
CASE <control expression> [OF]
CASE reply\$ OF

CAT ++ gives disk directory
CAT [<drive number>][<filename>]
CAT "shap.*"

CHAIN ++ load & run program on disk
CHAIN <filename>
CHAIN "menu"

CHANGE -- change text -"N" means no
CHANGE "<old text>",<new text>"
CHANGE "background","textbackground"

CHR\$ ++ gives the character specified
CHR\$(<numeric expression>)
PRINT CHR\$(num+128)

CLOSE ++ closes files
CLOSE [[FILE] <filename>]
CLOSE FILE 2

CLOSED ++ all proc/func variables local
PROC <procname>[(params)] [CLOSED]
FUNC <funcname>[(params)] [CLOSED]
PROC newpage(header\$) CLOSED
FUNC odd(number) CLOSED

CON ++ continue program execution
CON
CON

COPY -- copy a disk file
COPY <source name>,<target name>
COPY "test5","reportgen"

COS ++ cosine in radians
COS(<numeric expression>)
PRINT COS(number)

More ►

```

DISPLAY <proc/func name> [<filename>]
DISPLAY
DISPLAY update'record
DISPLAY readrec "proc.readrec"

```

DIV ++ division with integer answer
 <dividend> DIV <divisor>
result=guess DIV count

DO ++ used with FOR and WHILE
DO <statements>
WHILE ok DO
WHILE NOT error DO ask'question
FOR x=1 TO max DO show'item(x)

```
EDIT ++ list lines without indentations
EDIT [<line#>]
EDIT <range>
EDIT <proc/func name>
EDIT 20
EDIT 500-
EDIT pause
```

ELIF ++ short for ELSE IF condition
ELIF <expression> [THEN]
ELIF reply\$ IN "AEIOU" THEN

ELSE ++alternative IF struc statements
ELSE
ELSE

```
END ++ halt program & show message
END [<display message>]
END "All Done."
```

ENDCASE ++ end of CASE structure
ENDCASE
ENDCASE

ENDFOR ++ end of FOR structure
ENDFOR [**<control variable>**]
ENDFOR sides

ENDFUNC ++ end of function
ENDFUNC [<function name>]
ENDFUNC pause

More ►

COMAL 2.0 Keywords - continued

ENDIF ++ end of IF structure
ENDIF
ENDIF

ENDLOOP -- end of LOOP structure
ENDLOOP
ENDLOOP

ENDPROC ++ end of procedure
ENDPROC [<procedure name>]
ENDPROC show'item

ENDTRAP -- end of TRAP structure
 ENDTRAP
ENDTRAP

ENDWHILE ++ end of WHILE structure
ENDWHILE
ENDWHILE

ENTER ++ retrieve ASCII program lines
ENTER <filename>
ENTER "lst.testing"

EOD ++ End Of Data flag
EOD
WHILE NOT EOD DO

EOF ++ End Of File flag
EOF(*<filenum>*)
WHILE NOT EOF(infile) DO

ERR -- returns error # within HANDLER
 ERR
CASE err OF

ERRFILE -- returns file in use at error
ERRFILE
IF errfile=infile THEN

ERRTEXT\$ -- returns error message
ERRTEXT\$
PRINT errtext\$

```
ESC ++ stop key pressed flag
ESC
TRAP ESC<type>
UNTIL ESC
TRAP ESC+
```

EXEC ++ execute a procedure
[EXEC] <procname>[(<parameter list>)]
EXEC show'item(number)

EXIT -- use to leave LOOP structure
EXIT
IF file'exists(name\$) THEN EXIT

EXIT WHEN -- conditional exit to LOOP
EXIT WHEN <condition>
EXIT WHEN errors>3

EXP ++ natural log e to n
EXP(<numeric expression>)
PRINT EXP(number)

EXTERNAL -- external proc/funks
PROC<name>[<parm>][**EXTERNAL**<filnam>]
FUNC<name>[<parm>][**EXTERNAL**<filnam>]
PROC sum(section) *EXTERNAL* "ext.sum"
FUNC rec'size(name\$) *EXTERNAL* "ext.rec"

FALSE ++ predefined value = 0
FALSE
ok=FALSE

```
FILE ++ specifies a file is to be used
INPUT FILE <file#>[,<rec#>]: <varlist>
PRINT FILE <file#>[,<rec#>]: <vallist>
READ FILE <file#>[,<rec#>]: <var list>
WRITE FILE <file#>[,<rec#>]: <varlist>
OPEN [FILE] <file#>,<filename>[,<type>]
CLOSE [[FILE] <file#>]

INPUT FILE 2,text$
PRINT FILE outfile,count:name$
READ FILE infile,sub:name$,phone$
WRITE FILE 3,1:total'records
OPEN FILE 2,"scores",READ
CLOSE FILE infile
```

More ►

FIND "<text string>"
FIND "PROC"

```
FOR <var>:=<start> TO <end> [STEP <s>]
FOR sides=1 TO 4 DO
```

```

FUNC <name>[(<params>)] [CLOSED]
FUNC <name>[(<parm>)] EXTERNAL<filnam>
FUNC odd(number) CLOSED
FUNC call'answered EXTERNAL "ext.cl"

```

```
GET$(<filenum>,<# of characters>)  
text$=GET$(2,16)
```

GOTO <label name>
GOTO jail

HANDLER
HANDLER

```
IF <condition> [THEN]
IF <condition> THEN <statement>
IF errors>3 THEN halt
IF reply$ IN "yYnN" THEN
```

IMPORT <identifiser>[,<identifiser>]
IMPORT bold'char

```
<string1> IN <string2>
IF win$ IN guess$ THEN winner
```

```
INPUT [<prompt>:] <var list>[<mark>]
INPUT FILE <file#>[,<rec#>]:<var list>
INPUT prompt$: reply$;
INPUT FILE 2: text$
```

INT(<numeric expression>)
tally: +INT(number)

INTERRUPT [**<procedure name>**]
INTERRUPT flasher

KEYS
CASE KEYS OF

<label name>:
quick'quit:

LEN(<string expression>)
length=LEN(text\$)

```
:= or :+ or :-
total:-loses
```

LINK <filename>
LINK "pkg.francais"

```
LIST [<line#>]
LIST [<range>] [<filename>]
LIST <proc/func name> [<filename>]
LIST 10
LIST -33
LIST header "proc.header"
```

LOAD <filename>
LOAD "menu"

```
LOG(<numeric expression>)
PRINT LOG(number);
```

More ►

■■■■■

XX

SQR ++ gives square root
 SQR(<numeric expression>)
root=SQR(number)

STATUS ++ status of disk error channel
STATUS
STATUS

STEP ++increment FOR loop by this amount
STEP <numeric expression>
FOR *x*=1 **TO** *max* **STEP** 2 **DO**

STOP ++ halt program execution
STOP [<message>]
STOP "Now on line 350"

STR\$ -- converts number into string
STR\$(<number>)
zip\$=STR\$(number)

SYS ++ transfer control to machine code
 SYS(<memory address>)
SYS(828)

TAB ++ print spaces to specified column
TAB(*<column number>*)
PRINT TAB(col),name\$

TAN ++ gives tangent in radians
TAN(*<numeric expression>*)
PRINT TAN(number)

THEN ++ part of IF structure
THEN
IF ok THEN

TIME -- returns time in jiffies
TIME
TIME <set time>
PRINT TIME
TIME 0

TO ++increment FOR variable start TO end
 <start num> TO <end num>
FOR x=1 TO 4 DO

TRACE -- show how program got there
TRACE [*<filename>*]
TRACE "lp:"

TRAP -- disable stop key
TRAP ESC<type>
TRAP *part of ERROR HANDLER*
TRAP ESC-
TRAP

TRUE ++ predefined value of 1
TRUE
RETURN TRUE

UNIT ++ specify device (0.14 only)
UNIT <unit specifier>
OPEN FILE 255,"",UNIT 4,7,WRITE

```
UNIT$ -- returns the current unit
UNIT$
UNIT$="2:"
```

UNTIL ++ end of REPEAT loop
UNTIL <condition>
UNTIL reply\$="q"

USE -- use specified package
USE <filename>
USE dansk

```
USING ++formatted output
PRINT USING <format>: <var list>
PRINT USING "##> $###.##":x,cash(x)
```

VAL -- returns numeric value of string
VAL(<numeric string>)
age=VAL(reply\$)

VERIFY -check file against prog in memory
VERIFY <filename>
VERIFY "final"

WHEN ++ choice in CASE structure
WHEN <list of values>
WHEN "Jan","jan"

More ►

WHILE ++ start of WHILE structure
 WHILE <expression> [DO] [<statement>]
WHILE NOT EOF(in file) DO process

WRITE ++ write to a file
 WRITE FILE <fil#>[,<rec#>[,<offset>]]:<var>
 OPEN [FILE] <filenum>,<filename>,WRITE
WRITE FILE 2:name\$
OPEN FILE 3:"scores".WRITE

ZONE ++ tab interval
ZONE <tab interval>
ZONE
ZONE 5
old'zone=ZONE

0.14 NOTE:

-- after keyword means 2.0 only
 ++ after keyword means 0.14 and 2.0
 but 0.14 may not be fully implemented.
 Check COMAL Handbook to verify. ■

COLOR DUMP BUG

Terrance R on PlayNet provided us this fix to a bug he found in Ray Carters COLOR DUMP program on *TODAY Disk #9* (discussed on page 67 of *COMAL TODAY #9*):

Original line:
950 VALX=VALX*4

Fixed line:
950 VALX=(VALX-(VALXX*4))*4

HANDBOOK BUG

There is a mistake in the sample program listing for PROC on page 233 of the *COMAL Handbook*. The UNTIL line in the program should read:

```
UNTIL word$="0" 
```

David Powell sent in a short program to do simple statistical work on an array of numbers. The program is complete, but you could use the procedures in any other program without change.

```
// delete "statistics/demo"
// save "statistics/demo"
// by David Powell
//
PAGE
PRINT "This program will take a"
PRINT "a group of numbers and"
PRINT "calculate their average"
PRINT "and standard deviation"
DIM vector(100)
n#:=0
REPEAT
PRINT
PRINT "Enter element";n#+1
INPUT "(-999 to stop): ": x
IF x<>-999 THEN
    n#:=+1
    vector(n#):=x
ENDIF
UNTIL x=-999 OR n#=100
//
average:=mean(vector(),n#)
PRINT
PRINT "The average value of the"
PRINT "values entered is";average
PRINT
PRINT "The standard deviation is:";
PRINT std'dev(vector(),average,n#)
END
//
FUNC std'dev(REF array(),avg,m#) CLOSED
sumsq:=0
FOR i#:=1 TO m# DO
    x:=array(i#)
    sumsq:+(x-avg)*(x-avg)
ENDFOR i#
IF m#>20 THEN m#:-1
RETURN SQR(sumsq/m#)
ENDFUNC std'dev
//
FUNC mean(REF array(),m#) CLOSED
sum:=0
FOR i#:=1 TO m# DO
    sum:=+array(i#)
ENDFOR i#
RETURN sum/m#
ENDFUNC mean
```

2.0 Modem Update

by David Stidolph

In Denmark modems are quite expensive and their use is regulated by the government. The designers of COMAL 2.0 for the Commodore 64, UniComal ApS, could not test their cartridge with the modem. As a result, the COMAL cartridge does not use the correct routine when reading from the modem. PRINTing to the modem has always worked, but GET\$ will sit and wait for characters to come in, hanging your computer up in a wait state that even the STOP key will not break. This is not good!

The following procedure POKE's machine code into a free memory space that will correct this flaw in the COMAL cartridge. The low memory pointer to CHRIN (the routine COMAL uses when reading a byte from the keyboard or other device) is then changed to point to this new machine code. The code checks if COMAL is trying to get a character from the modem. If COMAL is, then the routine GETIN (used for modem input) is called. Otherwise the regular CHRIN routine is called. With this fix, we now have fast modem I/O.

```
PROC fix'modem CLOSED
  RESTORE modem'code
  FOR address:=&c86a TO &c875 DO
    READ num
    POKE address,num
  ENDFOR address
  POKE &0324,&6a
  POKE &0325,&c8
  //
modem'code:
  DATA &a5,&99,&c9,&02,&f0,&03
  DATA &4c,&57,&f1,&4c,&3e,&f1
ENDPROC fix'modem ■
```

Questions

TEXT SCREEN SPRITES

Question: Is it possible to show sprites on the text screen? - W Staneski

Answer: Yes it is possible to show sprites on the text screen, but not using the sprite keywords. CLOCK on Today Disk #3 has text screen sprites.

Reed Brown has these questions:

FREE and SIZE

Question: Is there any way to check remaining memory (like BASIC'S FRE(O))?

Answer: To check on the amount of free memory issue this command:

SIZE

In COMAL 2.0 you also have a function called FREE that can be accessed from within a running program:

USE system
PRINT free

TIME and JIFFIES

Question: Is there anyway to access BASICs TI (real time clock) variable?

Answer: The COMAL 2.0 command TIME is equivalent to the BASIC command TI. COMAL 0.14 does not have this built in, but you can use this function (followed by an example of how to call it):

```
func jiffies closed
  j:=256*256*peek(160)+256*peek(161)+peek(162)
  return j
endfunc jiffies

print jiffies ■
```



Now Available Through Aquarian Software

Gold Disk Series

Each Disk Contains:

- The Monthly Feature Program
- Programming Tutorials
- High Quality Games
- And Much More

Volumes 1 through 11 Now Available!!!

Volume 11 Features a C-64 Assembler

Gold Disk Series for 128
Coming Soon!

Only \$14.95 Per Disk*

* Plus Shipping and Handling

The Cataloger

The Ultimate Disk Cataloging System for the 64!

Features of The Cataloger V3.5A Include:

- * Loads directly from the disk itself.
- * Ability to change name of entry.
- * Fast — Uses relative files exclusively
- * Search, Sort and Print by any of 12 fields.
- * 1100-program (or disk) capacity per data disk.
- * All machine language.
- * Menu driven — very easy to use.
- * Works with one or two drives.

Only \$24.95

BobsTerm Pro

The Ultimate Terminal Software I

Upload / Download Supports Punter,
X-Modem, XON / XOFF, DC1 / DC2,
and Much More!

28.5 Byte Buffer with unmatched editing
abilities

- User Adjustable Parameters
- 10 Custom Character Sets
- Unlimited Phone Book Storage
- Programmable Macro Command Strings

Only \$59.95

MATRIX — NOW AVAILABLE!!

The Indispensable C-128 Utility / Starter Kit I

Use dozens of 128 features in the 64 mode:

- Numeric Key Pad
- Cursor Keys
- 80-Column RGB Output
- Many Other Special Function keys

One-Key Functions Include:

- 2 Megahertz "Fast Mode"
- One-Key Screen Dumps
- Full-Featured DOS Utility Menu

Other Features Include:

- Fast Loading
- Fast Copy For The 1571!
- Relocatable In Memory
- 100% Transparent to BASIC

Available Now **\$59.95**
For Only

Graphic Screen Exporter

A Universal Graphics Converter I

Converts Anything to Anything — Including:

Koala Pad Doodle
Flexdraw Print Shop
COMAL Paint Magic
CAD GEM Micron Eye

And Many Many More !!

The Most Versatile Graphics Utility Ever
Released for the Commodore 64 I

Only \$29.95

ALSO AVAILABLE:

OmiTerm.....\$19.95

Full-Feature Terminal at an Affordable Price!

Turbo Calc/64.....\$17.95

A great spreadsheet at an Unbelievable Price!

Tax Computation.....\$29.95

The friendliest tax package on the market.

Guitar Master.....\$49.95

A comprehensive musical instruction package

Fast Boot!.....\$14.95

Mike J. Henry's Fast Loader for 1541/MSD

Thriller Collection.....\$24.95

Seven intricate text adventures on one disk

Call or Write for Full Catalog I

CAD-GEM

Computer Assisted Design Graphic Element Manipulation

A Wire Frame CAD system for the C64 I
Input from Joystick, Track Ball, Light Pen or
Graphics Tablet

360 Degree Rotation in .1 Degree Increments
Scaling on a 64K x 64K, 2048 Mega-Bit Virtual
Screen
Independent Manipulation of 400 Objects (Points
or Lines)

You must see CAD GEM to believe it!
Demo Disk Available for \$3.00

\$89.95

MODEM MASTER

The Friendliest Commodore BBS Available

- Works with 1541 or MSD Dual Drive
- 300 / 1200 Baud Operation
- New Punter File Transfer Protocol
- Sub-Directories for File Transfer
- 250 User Capacity
- Accurate Clock / Calendar
- Printer Output
- Information Files
- "Old" E-Mail Deleted After One Week
- Set Up In Only 10 Minutes I

Only \$29.95

Total Software Development System

by Kevin Pickell
Now Available in the States I

Assembler/Editor — fast load, get, log and loadat; adds 38 new commands; full macro instructions;
allows 13-character labels; assembles to and from disk

Sprite Editor — 256 sprites in memory. view 64 at same time. works with keyboard, joystick or
trackball, animates sprites during design

Unassembler — create source code from any ML program

Sound Editor — create interrupt-driven sound effects

Character Editor — edit all characters. Screens to 255x64. Hi-res & Multi-color Character Sets

TSDS automatically includes sprites, characters, mattes and sound effects into source code!

Only \$39.95

128 Version Coming Soon I

Aquarian Software

P.O. Box 22184
Portland, OR 97222



To order, Call: (503) 654-2641
VISA & MasterCard Accepted



Add 3.00 S & H Per Order
(Add Additional \$2.00 for COD)
Canadian Orders Add 10.00 S&H
Allow 3-4 Weeks For Delivery

Write or Call for Full Catalog — Dealer Inquiries Welcome I

Disk Directories For Disk Sleeves

COMAL Today 1

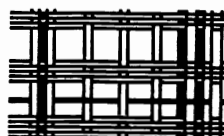
boot c64 comal	>-----<	dir'list	<u>88 Files</u>	<u>0 Blocks Free:</u>
c64 comal 0.14	information84mar	sky'catcher.l	color'funcs.l	link-program.l
-----	help-comal	sky'catcher	color'combo.l	link-program
>-----<	help-graphics	create'lander.l	color'combo	control.l
>error messages<	help-sprites	create'lander	hidescreen.l	new'program.l
> file <	>-----<	lander'sprites	showscreen.l	>-----<
>-----<	>comal programs<	lander.l	print'using.l	>the following <
comalerrors	>-----<	lander	print'using	>two programs <
>-----<	see'information	microscribble.l	note17.program	>are written in<
>file generator<	see'instructions	microscribble	val'demo.l	> basic <
-----	-----	paddletest.l	val'demo	> ----- <
generate errfile	big'letter.l	paddletest	square-a3	>do not load <
>-----<	big'letter	etchasketch.l	square-b3	>them into <
>auto boot prog<	bigletter/demo.l	etchasketch	design3	>comal. <
>-----<	bigletter/demo	light'pen.l	design4	>-----<
hi	logo'emulator.l	light'pen	square-c3	1541backup(free)
>-----<	logo'emulator	koala.l	squares3	single file copy
>seq data files<	dir'list	koala	sprite-a1	
			sprite-b1	

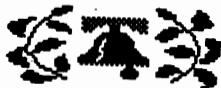
COMAL Today 2

boot c64 comal	information84mar	convert.l	<u>92 Files</u>	<u>17 Blocks Free:</u>
c64 comal 0.14	help-comal	disk'get2.l	proc1.l	rotpac.l
-----	help-graphics	func3.l	proc2.l	test lsf.l
>-----<	help-sprites	inside.l	put'char.l	test'clog.l
>error messages<	>-----<	meeting	reverse.l	test'cp'r.l
> file <	>comal programs<	meeting.l	screen'char.l	>-----<
>-----<	>-----<	note76.l	set'screen.l	>the following <
comalerrors	see'information	note79.l	sinename	>two programs <
>-----<	see'instructions	note80-1.l	sinename.l	>are written in<
>file generator<	-----	note80-2.l	front'page	> basic <
>-----<	8023p'options4	note81-1.l	sprite'editor21	> ----- <
generate errfile	8023p'options4.l	note81-2.l	stars1.l	>do not load <
>-----<	background.l	note96.l	stars2.l	>them into <
>auto boot prog<	benchmark64.basi	note97.l	string	>comal. <
>-----<	benchmark64.coma	page 15 sample.l	string.l	>-----<
hi	benchmark64.l	page.l	vt-52.v4	1541backup(free)
>-----<	border'color.l	pens'color.l	vt-52.v4.l	single file copy
>seq data files<	clear'collisn.l	pentagram.l	plexpac.l	
>-----<	convert	plottext.l	vecpac.l	
			matpac.l	

COMAL Today 3

boot c64 comal	help-comal	getbackground.l	<u>99 Files</u>	<u>50 Blocks Free:</u>
c64 comal 0.14	help-graphics	getborder.l	page.l	spritestat.l
-----	help-sprites	getpen.l	penstat.l	spritexcor.l
>-----<	-----	getpencolor.l	pie'chart'maker	spritexsize.l
>error messages<	>comal programs<	getspritecolor.l	pie'chart'print	spriteycor.l
> file <	>-----<	getturtlesize.l	plot'char.l	spriteysize.l
>-----<	bit'map'print.l	graphicstate.l	plot'char/demo	turtlestat.l
comalerrors	circle.l	heading.l	polygon.l	user'error/demo
>-----<	clock	hidescreen.l	quicksort'number	xcor.l
>file generator<	comal'dump.asm	hidescreen2.l	quicksort'string	ycor.l
generate errfile	comal'dump.bas	jiffies.l	random'file/demo	>-----<
>-----<	convert(new).l	keyword'print	randomize1.l	>the following -
>auto boot prog<	create.l	load'demo	randomize2.l	>basic program -
>-----<	curcol.l	load'demo2	save'demo	>is a single -
hi	currow.l	load'screen.l	save'screen.l	>drive copier. -
>-----<	cursor.l	load'screen2.l	set/readtime.l	>-----<
>seq data files<	demo	moire.hrg	setx.l	>do not load it-
information84jun	dir'manipulator	mount.l	sety.l	>while in comal-
	drawletter.l	obj'load.l	showscreen.l	>-----<
		obj'save.l	showscreen2.l	1541backup(free)
			showsprite.l	





COMAL Today 4

boot c64 comal
c64 comal 0.14
comalerrors
hi
abc.sprite
alpha1/gen
alpha1.dat
alpha2/gen
alpha2.dat
comal'bug-a
comal'bug-b
dec'to'hex/demo
diamond

disk'protector
disk'talk'examp1
disk'talk'examp2
dodge'em
dump'1525
dump'big'epson.l
dump'nec8023a
dump'prowriter
dumpscreen.proc
dumpscreen/demo
dumptext1525.l
dynam'data
find'string/demo

gutenberg'shell
gutenberg'ademo
gutenberg'bdemo
hex'to'dec/demo
joystick.proc
lock'lower.proc
lock'upper.proc
logical'ops.func
ml'string/demo
music'all/demo
num-string/demo
opt'triangle
optical'hexagon

65 Files

paddle.proc
polyspirals
prime/demo
ram'errors
random'plot
rocket.sprite
screen'location
scroll'down/demo
seq'print
spiralateral
stars
takeoff/demo
tiles

0 Blocks Free:

tri'hex/demo
two'drive'copier
two'drive'instru
unlock'case.proc
wall'clock
wandering
>-----<
>the following <
> program is <
> written in <
> basic <
>-----<
1541backup(free)

COMAL Today 5

hi
comalerrors
>comal programs<
alarm'system
boxtree
color'mix
connected'boxes
correlator
datacollision
disk'editor
dog/cat
draw'sine'wave
exmpl'bar'graph
expand'memory
grade'distribute
guess'it
hypotenuse
identify

inventoryprogram
leap'year
let&using'exmpl
magic'fruit
many'patterns
many'stars
ml'setup
names'printout
oki92'hi
oki92'screen'io
pitfall'harry
polar'daisy
polar'long
queens(ver 0.14)
recursion'exmpl
rotated'ovals
show-stopper
sign'language

spritecollision
turtle/demo
wage'demo
xploded'pie
>---data files---<
bigdump'nec.obj
bigdump'nec.src
dat.bwv783
fingera
fingerb
fingerc
fingerd
fingere
fingerf
fingerg
fingerh
fingeri
fingerj

89 Files

fingerk
fingerl
fingerm
fingern
fingero
fingerp
fingerq
fingerr
fingers
fingert
fingeru
fingerw
fingerx
fingery
fingerz
inventory
oki92.dump.obj

1 Blocks Free:

>---listings---<
base'convert.l
clear'keys.proc
func.modemget2.0
graphs.l
ml'procs
save'screen.proc
test'signal.l
>---screens---<
1st 80 kanji.hrg
2nd 80 kanji.hrg
>-2.0 programs-<
all'at'once2
memory'peeker2.0
>--benchmarks--<
sieve.comal.l
sieve.basic

COMAL Today 6 Front

hi
menu
>---programs---<
1525'screen'dump
aprilfool
art
boot'data'base
boot'fit'it
bounce'ball.14
bowling'score
comal'keypad.14
create'lib'data
data'base'mgr

dbase14
draw'daisy
expand'ram
find
fit'it
grades
label
librarian
magic
mail'label
microscope'quiz
mindy's-demo
pinwheel

playnet
print'2'col'dir
print'calendar
print'directory
quadratic'root
save'color'data
save'color'pokes
save'error'file
star'80
view'color'data
view'color'pokes
view'logo
>---procedures---<

63 Files

>-&'functions---<
buffer.proc
dir.l
file'exists.func
fx-80'cmds.proc
joystick.proc
load'obj.proc
paddle.proc
plottext.proc
repeat'key.proc
restore'lbl.proc
>---data'files---<
---error-mess---

6 Blocks Free:

box&circle.pic1
box&circle.pic2
comdmppmod.mem
dance.dat
hrg.world'map
lib.dat
load/save.mem
microscope.dat
music'player.mem
phone'data
phone'dict

COMAL Today 6 Back

hi
>---programs---<
battleship
bounce'ball
calculart
check'all'cards
class'labels
comal'keypad'2.0
comal'clock
create'silicon
disassembler

dog/cat
doodle'to'2.0
draw'molecules
f-key'overlay
fit'it'2.0
gemini'dir'print
grading
hex'dump
label'it
letter'shell
lightpen'demo1

lightpen'demo2
log'program
magic'voice'demo
polar'roses
poster'down
poster'right
shadow'letters
shell'sort
song'editor
song'player
>---external---<

54 Files

ext.qtr'grade
ext.sem'grade
>---procedures---<
proc.buffer
proc.fx-80'cmds
proc.gem'bigdump
proc.gemini'dump
proc.magic'voice
proc.modem'work
proc.olivetidump
proc.repeat'keys

34 Blocks Free:

>---data'files---<
bat.0.14'to'2.0
bat.dual'stuff
bat.edit'prog's
bat.normal
dat.logfile
dat.silicon
hrg.front'page
sng.polka/acc
sng.polka/str

COMAL Today 7 Front

boot c64 comal	1520/0.14demo7	1541'alignment	<u>51 Files</u>	<u>9 Blocks Free:</u>
c64 comal 0.14	1520/0.14demo8	1541'alignment1	txt'dump'ctl-p	information.dat
hi	1520/0.14demo9	1541'alignment2	>--procedures--<	move'basic-\$9000
menu	1520/0.14demo10	add'errors	1520/driver.proc	sample.dat
>---programs---<	1520/0.14demo11	comal'program	blankscreen.proc	sample.dat-1
1520/0.14demo1	1520/0.14demo12	eliza	get'protect.func	sample.dat-2
1520/0.14demo2	1520/0.14demo13	freeway	graphics.proc	sample.dat-3
1520/0.14demo3	1520/0.14demo14	gem10x'card/a	menu.proc	sample.dat-4
1520/0.14demo4	1520orbit'circle	input'on'input	show'error.proc	
1520/0.14demo5	1520sphere'plot	mailing'list	use'extend.proc	
1520/0.14demo6	1520test'print	sort'all	>---data-files--<	
			-error-messages-	

COMAL Today 7 Back

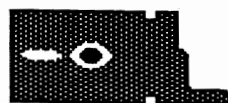
hi	1520demo14	diff'equations	<u>73 Files</u>	<u>19 Blocks Free:</u>
>---programs---<	1520freeway	disassembler	perform'demo1	proc.restore
1520demo1	1520orbit'circle	disk'editor	perform'demo2	proc.use'extend
1520demo2	1520sphere'plot	draw'heart	polygons	>---data-files--<
1520demo3	1520test'print	eliza	prnt'lrg'chars	bat.1520'40-list
1520demo4	80'column'demo	find'radical	read'directory	bat.1520'80-list
1520demo5	ahl's-benchmark	flow	sound.sample	bat.link-802dump
1520demo6	batch'example	freeway	test'external	dat.information
1520demo7	batchfile'maker	gem10x'card/a	>---procedures--<	font.80column
1520demo8	calculart2	input'on'input	func.get'protect	font.roundset
1520demo9	check'all'carts	joy'cursor	func.menu	font.typeset
1520demo10	cloud'flux	k-s'stat'test	func.modem'get\$	>---external---<
1520demo11	comal'program	make'data'stmts	proc.1520driver	ext.double
1520demo12	convert'listing	make'object'file	proc.blankscreen	ext.make'double
1520demo13	demo.fkeys	multiplication	proc.comal-802	
			proc.load'font	

COMAL Today 8 Front

boot c64 comal	music'compiler	sound.proc	<u>74 Files</u>	<u>6 Blocks Free:</u>
c64 comal 0.14	music'demo	use'sound.proc	shap.'f'	shap.harry05
hi	seq'to'speed	>---data-files--<	shap.'k'	shap.harry06
menu	sidmonitor	-error-messages-	shap.'l'	shap.harry07
>---programs---<	soundex	dance'hours.dat	shap.'m'	shap.harry08
art'nouveau	speed'to'seq	dance'hours.sng	shap.'n'	shap.harry09
boot'dir'editor	sprite'converter	dir'editor.mem	shap.'o'	shap.harry10
colorwheel'demo	sprite-sample4	dutch'errors	shap.'r'	shap.harry11
concentration	trees	help.dat	shap.bat1	shap.men0
create'bats	view'koala	information.dat	shap.bat2	shap.men1
depreciation	view'sprites	music'player.mem	shap.bat3	shap.men2
directory'editor	>---procedures--<	musicroutine.src	shap.harry00	shap.queen
disk'edit/protct	comal'music.proc	>---shape-files--<	shap.harry01	shap.santa0
display'seq'file	loadshape.proc	shap.'a'	shap.harry02	shap.santa1
illusion	saveshape.proc	shap.'c'	shap.harry03	shap.santa2
			shap.harry04	

COMAL Today 8 Back

hi	db'squash	sprite-sample4	<u>74 Files</u>	<u>17 Blocks Free:</u>
>---programs---<	font'sprite	stack'space	proc.cdate	dat.game'names
beams	fun'print	star-zamara	proc.change'dev	dat.information
color'wheel'demo	illusion	statistics/demo	proc.directory	db'data
concentration	interrupt'demo	test'disk	proc.scroll'down	db'help.def
create'bats	make'data	test'pattern	proc.sirene	db'help.lab
cross'reference	make'err'english	trees	proc.stand'dev	db'help.rpt
db'boot	make'font	word'game	proc.toggle'keys	db'name
db'define	morgage	view'sprites	proc.type	pkg.basic
db'help	run'basic'prog	>---procedures--<	proc.window'down	pkg.dutch
db'labels	scroll'message	func.convert'bas	proc.window'up	src.basic
db'maintenance	seq'to'speed	func.get'input	>---data-files--<	>---shape-files--<
db'menu	soundex	func.jdate	bat.demo	shap.bat1
db'report	speed'to'seq	proc.alarm	bat.loop	shap.bat2
db'sort	sprite'converter	proc.average	bat.sample	shap.bat3
			comal	





COMAL Today 9 Front

boot c64 comal
c64 comal 0.14
hi
menu
ml.sizzle
comalerrors

-copy the above-
-files together-
-to other disks-
-for the sizzle-
- loader -

>---programs---<
1520flag'day.14
1520max'print.14
1520polygons.14
1541'alignment
1541'align'1
1541'align'2
create'sizzle
metamorphose
program'3
program'4

seq'to'speed
speed'to'seq
structure'prg'1
structure'prg'2
structure'prg'3
structure'prg'4
structure'prg'5
structure'prg'6
waves'keybd'demo
- procedures -
- and -

56 Files

- functions -
1520/drv.proc
decimal.func
drive'type.func
dump'1525.proc
dump1520.proc
epson'cardg.proc
load'sizzle.proc
read'errors.proc
showtable.proc
zerotable.proc

0 Blocks Free:

- data file -
information.dat
-basic program -
fast'boot.bas

COMAL Today 9 Back

hi

- programs -

1541'alignment
clue
convert'num
dates&julian
direct'con
gemini'colordump
icon'maker
infantry
magic'paint
oki92'test
prog'ram
program'1
program'2

rod
roll'over'basic
seq'to'speed
show'errors
single'file'copy
speed'to'seq
tank'animate
viewport
waves'demo
yahtzee
- functions -
- and -
- procedures -
func.decimal
func.drive'type

func.last

func.mean
func.random'size
func.rms
func.sdev
func.sigma
proc.1520plotter
proc.convert1
proc.convert2
proc.convert3
proc.epson'cardg
proc.graph'keys
proc.show'names
proc.showproc
proc.trunc
proc.window'down
proc.window'up

81 Files

- data file -
dat.information
- packages -
pkg.first'last
pkg.oki92
src.first'last
src.oki92
-files used by -
- 'rod' -
- do not load -
roderigue

0 Blocks Free:

problems
shap.down'rod
shap.lt'rod
shap.rt'rod
shap.up'rod
- printshop -
- pictures -
cookie'monster
donald'duck
garf.head
goofy

COMAL Today 10 Front

boot c64 comal
c64 comal 0.14
hi
menu
ml.sizzle
comalerrors

- comal 0.14 -
- programs -

aim
boot'tutor
crg14.compactor
crg14.filewriter
crg14.viewer
curves
design
dvorak.14
hi-lo'game

kilroy
missing'letters
num'to'word
random'sampler
telephone
tutor'amnesia
tutor'remember
view'font/demo
walker
- basic fonts -
basic fonts
set.art deco.b
set.roman.a
set.tech2.a
- compacted -
- pictures -

compact pix
bbs.crg
blackcomal.crg
calvin.crg
cug.color.crg
fragile.crg
goofys car.crg
guard.crg
guards.crg
loon.crg
natalie.crg
train1.crg
- bitmaps -
directory
santa.ct10.hrg
school room.hrg

92 Files

- functions -
- and -
- procedures -
bubblesort.proc
bubblesort2.proc
bubblesort3.proc
change'8to9.proc
compact'14.proc
dir.proc
exchange.proc
load'comp.proc
load'font.proc
quicksort.proc
restore'scn.proc
select'sort.proc
set'text.proc
shuffle'in.proc

2 Blocks Free:

spritecolor.func
- data files -
dbase.dat
phone.dat
src.compactor.14
-basic programs-
- do not load -
- from comal -
backupdisk.basic
copyfiles.basic
single file copy

COMAL Today 10 Back

hi

- comal 2.0 -
- programs -

aim
atom'happy
boggle
comal'ace
compare
copy
copy/compare
crg2
cubes
curves
design
font editor
kilroy

meta demo 1
missing'letter
movie viewer
note'teacher
num'to'word
random'sampler
show'drives
spelling'game
telephone
text'effects
transposition
walker
- data files -
film.big movie
missing.dat
names.ran

phone.dat

- fonts -
font.outline
font.repton
font.roman fancy
- functions -
func.drive'type\$
func.loadcompact
func.random'size
func.savecompact
func.spritecolor
- packages -

86 Files

pkg.bitmap
pkg.cmon
pkg.compactor
pkg.eps'grph8009
pkg.eps'grph9000
pkg.exeq
pkg.finchutil
pkg.meta
pkg.rotate
- procedures -
proc.bubblesort
proc.bubblesort2
proc.bubblesort3
proc.change'8to9
proc.dump'laser
proc.exchange

6 Blocks Free:

proc.get'drives
proc.link'font
proc.link'meta
proc.quicksort
proc.reveal
proc.select'sort
proc.shuffle'in
- shape files -
shap.comal today
shap.house
shap.question
shap.tree

User Group 1

boot c64 comal	mail'label'hs	c'curve	<u>57 Files</u>	<u>230 Blocks Free:</u>
c64 comal 0.14	---procedures---	circles'kev	graf1	rnd'bounce
comalerrors	title'page.proc	clock	graf2	serpent
hi	shell'sort\$.proc	cones	graf3	spiralateral
>-----<	header.proc	curve	graf4	spiro.plain
--screen dump---	-printer utility	design 2	graf5	starry night
1525 screen dump	seq'print	design 3	graf6	the'thing
--disk copiers--	-demo programs--	design 4	grid	turtlestick 6
sd2 copier	3d'cube	diamond	hanoi	turtlestick 7
sd2'copy&label	aprilfool	dragon	hilbert	who
--label makers--	beeper	eight boxes	life	
label	bounce	etch-a-sketch	p'circle	
			percent gain	

User Group 2

boot c64 comal	alpha2.dat	circle.proc	<u>42 Files</u>	<u>110 Blocks Free:</u>
c64 comal 0.14	gutenberg'ademo	title'page.proc	flowers	polymusic
comalerrors	gutenberg'bdemo	----programs----	four circle	spiral'star
hi	--screen dumps--	basic'to'comal	hamburger	squirrel mod
-feature program	bit'print'epson	coin flip	hourglass	star'power
gutenberg'shell	1525 pixel dump	comal art	lifer	starwatch
alpha1'gen	---procedures---	ct.header	lissajous	tabby
alpha1.dat	x-ycor.proc	curvette	new'house	
alpha2'gen	heading.proc	find'string.demo	nh3.exc	
			pinwheel	

User Group 3

boot c64 comal	nec ml dump	-demo programs-	<u>46 Files</u>	<u>109 Blocks Free:</u>
c64 comal 0.14	ml'dump.obj	a maz'in basic.b	circles	shapes
comalerrors	---procedures---	a maz'in comal.c	color swirl	sin'on'the'side
hi	colors.proc	a maz'in simon.s	comal promo	spiral'cir
-twin 1541 copy-	--sprite files--	angry.dragon	crazy quilt	spirograph
tddcfc	abc.sprite	another'moire	fft'model	spotty
tddcfc.inst	rocket.sprite	atari graphics	key draw	tiles
--screen dumps--	---utilities----	bagel	music'all'.demo	
prowriterdump.l	dynam-data-8	beep/gong.demo	pascalstrekant	
nec8023 scrndump	84expenditures	chris'star	polar	
			rnd color sqs	

User Group 4

boot c64 comal	---utilities----	arabesque2	<u>51 Files</u>	<u>105 Blocks Free:</u>
c64 comal 0.14	dumpscreen.1525	arabesque3	fanfare	son'of'pinwheel
comalerrors	terminal	arabesque4	flasher	spiral'sqr
hi	comalerror'gen	arabesque5	keno.game	sprite'circle
-----	---procedures---	arabesque6	kenoboard	starwars
--applications--	accept'demo	arabesque7	name game	supersketch
----programs----	hersh demos	checkerboard	perm'game	weirdness
record keeper	str\$.proc	clue	print boxes	window
label-ab	val2.proc	comal 2.02 boot	rectan	
budget	-demo programs-	dizzy turtle	rectangulus	
address	arabesque1	drum	scale	
			son of moire	

User Group 5

boot c64 comal	design.dat	city'scape	<u>53 Files</u>	<u>116 Blocks Free:</u>
c64 comal 0.14	flake.dat	color mix	flurry	polygoncrazy
comalerrors	e.dat	color'pic'loader	fourier sq	probability
hi	f.dat	cube designer	graphics demo	random'show
--screen dumps--	g.dat	dog/cat	haiku	readsprite/demo
neccomaldump	-demo programs-	dot.and.line	house	strs retirement
txt.screendump.l	arabesque8	drawshape.demo	kaleidoscope	therapy
---functions----	arcs.demo	drumette	moon phase	turtlestick8
asc.func	card'dealer	envelope'print	moving'turtle	>load from basic
scr2petsci.func	circle'maker	fast'circle	music'synth'eric	midwestern 4.0
---data files---	circle2.demo	fillbox	pauls math'add	
			poly'circles	





User Group 6

boot c64 comal
c64 comal 0.14
comalerrors
hi
----featured----
----program-----
game'unfin
---data files---
--do not load---
lakeside
e
f

g
tbsk
cmk
kbit
farmbit
farmtbs
farmcm
treebit
treetbs
treecm
help1
cbit

ctbs
ccm
help2
---procedures---
beep.proc
cat'.proc
tod.proc
wait'n'go.proc
cursor.proc
---utility-----
---programs-----
dir'print'nec

56 Files

imp.dump
pretty printer
disk'editor
dual epon.dump
expand'comal
-demo programs-
city patterns
comaldice
flower'judy
fractal
galactic news
orbit'circle

51 Blocks Free:

pitfall harry
polyhedron
queens
random keyframes
sphere'plot
traffic.light
variable boxtree
yarn'art1

User Group 7

boot c64 comal
c64 comal 0.14
---error-mess---
hi
>--data files--<
>-do not load--<
adv guess it
amort
guess it
guess it inst
language notes

microscope.dat
>---education---<
boot guess it
chill
geometry
grades
humidity
microscope quiz
perfect numbers
>-----games-----<
battleship

bounce'ball
codemaker
dots
mastermind'word
score keeper
>-applications-<
biweekly savings
bowling score
college ed savin
find
gem10x'lister

55 Files

loan
loan'chart
value of investm
>-----demos-----<
2'sine
art'color
beep'2
daisy color demo
fast'rectan
green boxtree
obells

7 Blocks Free:

one'more'circle
ovals
pinwheel
polar daisy
quad
rainbows
rotated ovals
spirograph'new
star 80
tangent circles
yarn'art'2

User Group 8 (England)

bootcomal
c64 comal 0.14
em
comalerrors

education

demo'select'sort
selection'sort.l
demo'bubblesort
bubblesort.l

demo'insert'sort
insertion'sort.l
introd'quicksort
demo'quicksort
quicksort'vert.l
sort'timer'prg
rnd'name\$1000
demo'bin'search
binary'search
for'loop'part'1
for'loop'part'2

utilities

load'dump'epson
dump'epson
dump'mps801
uptols.l
three'd.l
geometry.l

database

51 Files

filing
1541 database

engineering

section
reliability

instructions

11 Blocks Free:

read'about'disk
about'disk
file'to'print
letter

v1.1 06/02/85

User Group 9 Fro.it (COD)

boot cod
go
title
<<< programs >>>
<<< basic >>>
doodle loader
nutcracker
ravics term
<< comal 0.14 >>

boot data base
dump.1525.big
simulate playnet
<<< comal 2.0 >>>
mail'list
joy cursor
<print shop pix>
shell
palm

waves
bunny
clavier
>notes
small note
garfield
odie
garf.head
<< data files >>

41 Files

<< don't load >>
-machine lang
load/save.mem
music.dat
player.obj
hrg.cover
data'base'mngr
dbase14
nutcr2

1 Blocks Free:

nutcr3
nutcr4
<< doodle pix >>
ddcomal calvin
ddcabbage patch

User Group 9 Back (COD)

menu
articles
language
mainmenu
prg'inst
programs
sig'news
standard
advertis
help cod
want'ads

table of content
colin does sf
flash
moving up to 2.0
on balans chair
open letter
party quiz
playnet part 1
playnet part 2
randomthoughts
wild cards

your niche
1541 tips
april preview
commodore ecks
directory one
directory two
other groups
our group
our library
our newsletter
our officers

52 Files

prez page
to n/l editors
butterfield tape
capt comal visit
comal classes
hooking in
logo & pilot
playnet
user group disks
vic 20 list
data base inst

1 Blocks Free:

doodle loader
dump.1525.big
joy cursor
mail list
nutcracker
print shop pix
ravics'term
simulate'playnet



User Group 10

boot comal	load/save.mem	- comal 0.14 -	<u>84 Files</u>	<u>5 Blocks Free:</u>
c64 comal 0.14	mirrow.dat	- programs -	- utilities -	sunflake
hi	phone data	- applications -	find'load'addr	yarn'art'3
menu	txt.term14.inst	code trainer	load'color'scrn	-the following -
- data files -	---procedures---	drunkardwalk	mirrow writer	- requires the -
- ldon't load! -	- enter only -	invoice	p1090 char edtr	- 1520 plotter -
-----	- ldon't load! -	linear'regressn	print show.nec	formatter.1520
-error-messages-	-----	nursery libs	- graphics fun -	sunflake.1520
directory	disk'get'improv	phone log	-----	-----
dump.nec	plotter.procs	pulse rate	art	comal user group
spi-main	two'tone.proc	rhyming speller	dots.game	6041 monona drv.
alpha1.dat	colr'bar.proc	sieve'eratosthe	koala doodler	madison,wi 53716
e & jim v.hrg	val.lp.proc	sim'equations	queens	(608)222-4432
hrg.air force	min'max.func	sundial	son o spirograph	-----
information.dat	-----	terminal.14	-----	-----

User Group 11 (2.0)

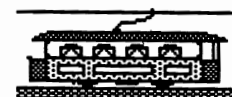
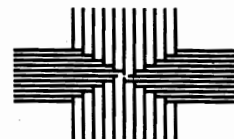
- programs -	- do not load -	dsp.circles	<u>96 Files</u>	<u>1 Blocks Free:</u>
bitmapdoodle&2.0	dat.bwv779	dsp.hobie	-copy the files-	cardv85
color'hanoi	dat.bwv781	dsp.hobie2	-below to a new-	categories
plot'a'function	dat.bwv783	dsp.jane	- disk -	exnames
roman'numerals	dat.bwv786	dsp.my'moire	-----	hg85
the'memory'game	dat.bwv794	dsp.turtlejoy	-master program-	inc85
- music and -	dat.bwv801	dsp.turtlejoy2	main'menu	incnames
- graphics -	mem.raster	dsp.wild	-----	jb85
-----	- 1520 plotter -	ext.circles	- sub-programs -	post85
dizzy'turtle	-----	ext.hobie	create.categorys	- the above -
drawto	1520draw'house	ext.hobie2	do'posting	- files should -
kaleidoscope	kaleidoscope1520	ext.jane	expense'record	-be transferred-
paint'by'letter	-----	ext.turtlejoy	income'record	- together -
raster'scanner	-external proc -	ext.turtlejoy2	init'expense	-----
sampler	- examples -	ext.wild	init'income	-----
- data files -	main'program	-2.0 accounting-	init'post	-----
-----	-----	system -	- data files -	-----

Programmers Paradise

boot c64 comal	missing'letters	- data files -	<u>86 Files</u>	<u>1 Blocks Free:</u>
c64 comal 0.14	playnet/quantum	- do not load -	- pictures -	copyfiles.basic
hi	random'sampler	-----	compact pix	-for more info -
menu	structure'prg'1	microscope.dat	calvin.crg	- send sase to -
ml.sizzle	structure'prg'2	missing.dat	chip.crg	-----
comalerrors	structure'prg'3	names.ran	fragile.crg	- comal users -
- comal 0.14 -	structure'prg'4	paradise.txt	helo.crg	- group, usa -
- programs -	structure'prg'5	phone.dat	loon.crg	- 6041 monona -
-----	structure'prg'6	programnames.dat	-----	- drive #202 -
aim	telephone	-----	-the following -	- madison, wi -
crg14.viewer	view'font/demo	-character sets-	- two programs -	- 53716 -
curves	walker	basic fonts	- can copy the -	-----
depreciation	-----	set.art deco.b	- disk or just -	-(608)222-4432 -
design	-all files that-	set.roman.a	- single files -	-----
freeway	-follow cannot -	set.tech2.a	-basic programs-	-----
illusion	-be loaded from-	-----	-----	-----
microscope'quiz	- comal -	- compressed -	backupdisk.basic	-----

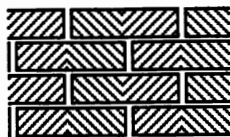
Newsletter Articles

compare-chart	double'disk	phone.list	<u>49 Files</u>	<u>10 Blocks Free:</u>
what-is-comal	filename-conv	walker.article	ratings	early'notes
questions-all	how'to'type'in'p	walker.list	valgol	questions9
letters-all	comalinschool	aim.article	ifthen	europe
newsletter	top10books	aim.ppc	get	questions9b
schools	how'to'start	design.article	how'to'draw-prt1	languages
look'back	how'to'do'it	design.list	how'to'draw-prt2	beginning'0.14
real'beginner	missing.article	copyfiles-new	finding'drives	strings
copyfiles	missing.list	book-text	questions10	questions
backupdisk	phone.article	quotes	expert'system	wp-files
-----	-----	-----	metathink	-----





B J C



Beginning COMAL

```

boot c64 comal >-----< exe72
c64 comal 0.14 exe17 exe73
>-----< exe18 peanuts
>error messages< exe19 exe69
>-----< exe110 exe73b
comalerrors exe22 bignuts
>-----< exe23 exe82
>file generator< exe31 exe91
>-----< exe32 exe94
generate errfile exe34 exe101
>-----< exe36 exe105
>auto boot prog< exe41 addition
>-----< exe42 unclxmas
hi discount exe122
>-----< exe45 exe123
>seq data files< exe46 exe131
>-----< exe51 exe141
information84mar exe52 test141
help-comal exe53 auntie
help-graphics oddeven festivals
help-sprites exe71

```

103 Files

```

doctor
eggs
wheel
enrol
correct
entermarks
report
hannibal
wordgame
xmascards
xmasfile
enternames
enteroptions
newshop
writeoffer
addrpr
exhibitoptions
> ----- <
> data files <
> ----- <
markbooks

```

87 Blocks Free:

```

xmas
options
offer
addresses
>-----<
> end of <
>beginningcomal<
>-----<
>the following <
>two programs <
>are written in<
> basic <
> ----- <
>do not load <
>them into <
>comal. <
>-----<
1541backup(free)
single file copy

```

Foundations Disk

```

boot c64 comal program5 program31
c64 comal 0.14 program6 program32
>-----< program7 program33
>error messages< program8 program34
>-----< program9 program35
comalerrors program10 program36
>-----< program11 program37
>file generator< program12 program38
>-----< program13 program39
generate errfile program14 program40
>-----< program15 program41
>auto boot prog< program16 program42
>-----< program17 program43
hi program18 program44
>-----< program19 program45
>seq data files< program20 program46
>-----< program21 program47
information84mar program22 program48
help-comal program23 program49
help-graphics program24 program50
help-sprites program25 program51
>-----< program26 program52
program1 program27 program53
program2 program28 program54
program3 program29 program55
program4 program30 program56

```

144 Files

```

program57
program58
program59
program60
program61
program62
program63
program64
program65
program66
program67
program68
program69
program70
program71
program72
program73
program74
program75
program76
program77
program78
program79
program80
program81
program82

```

105 Blocks Free:

```

program83
program84
program85
program86
program87
program88
program89
program90
program91
program92
program93
program94
program95
program96
program97
program98
program99
program100
program101
program102
program103
program104
program105
program106
program107
program108

```

Foundations Disk - continued

```

program109 program112 program115 program118 program121
program110 program113 program116 program119 telefon
program111 program114 program117 program120

```

Best of COMAL 0.14

```

boot c64 comal color'combo print'error'mesg
c64 comal 0.14 daisy'color'demo queens
hi dir'manipulator random'show
ml.sizzle dir'printer3 sky'catcher
comalerrors disk'editor split'screen
>-----< exploded'pie sprite'editor21
- comal 0.14 - flurry turtle'tutor
- programs - galactic'news wall'clock
>-----< guess'it wind'chill
april'fool gutenberg
arabesque4 hilbert
auto'directory ink'blot
bounce'ball magic'fruit
chris'star music
clue polymusic

```

73 Files

```

-to merge with -
- your own -
- programs -
file'exists.func
shift.proc
>-----<
- data files -
- do not load -
>-----<
- procedures -
- and -
- functions -
>-----<
- use- enter -
alpha2.dat
design.dat
flake.dat
raster.mem

```

7 Blocks Free:

```

>-----<
-for more info -
- send sase to -
>-----<
- comal users -
- group, usa -
- 6041 monona -
- madison, wi -
- 53716 -
>-----<
- or call -
-(608)222-4432 -
>-----<

```

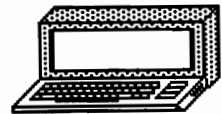
Utility Disk 1

boot c64 comal	dumpscreen/demo	copy&label	58 Files	1 Blocks Free:
c64 comal 0.14	file'to'print	dir'listner	vt-52.v4	print'seq'files
hi	8023p'options4	dir'manipulator	>----other-----<	quicksort'number
comalerrors	>---sprites-----<	disk'get/demo	basic'to'comal	quicksort'string
-----	abc.sprite	disk'protector	color'combo	remove comments
generate errfile	sprite'editor	file'to'screen	dec'to'hex/demo	save'screen.l
ram'errors	>----sound-----<	two'drive'copier	dynam-data-8	turtle/demo
>---printer-----<	beeper.l	two'drive'instru	find'string/fast	>the following <
dump'1525	fanfare	utilities	find'string/full	>program is in <
dump'big'epson.l	music/demo	view'directory	formatter2	>basic <
dump'nec8023a	>---disk access-<	>---modem-----<	hex'to'dec/demo	1541backup(free)
dump'prowriter	auto'directory	terminal	load'screen2.l	
			pie'chart'maker	



Font Disk Front

boot	font.fancy	font.round	56 Files	10 Blocks Free:
font editor	font.french	font.russian	font.warbot	- group,usa -
comal fonts	font.gothic	font.san quentin	-----	
-----	font.greek	font.script	- multicolor -	- may not be -
- comal fonts -	font.greek lang	font.streamline	- fonts -	- copied or -
-----	font.hebrew	font.tech1	-----	- placed in -
font.80column	font.italic.asci	font.tech2	font.mc.shaded	- user group -
font.art deco.pb	font.music notes	font.thick chars	font.mc.shaded1	- libraries -
font.ascii	font.outline	font.thin europe	font.mc.square	-----
font.colin	font.repton	font.type	- font disk #1 -	
font.computer	font.roman	font.typeset	-copyright 1985-	
font.d&d1	font.roman fancy	font.underline	- comal users -	



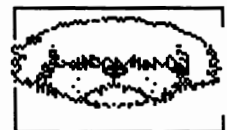
Font Disk Back

boot	- program -	set.art deco.b	90 Files	5 Blocks Free:
font viewer	-----	set.colin.a	set.sanquentin.a	view'font/demo
singlecopy	rotate.pkg.demo	set.colin.b	set.script.b	-----
information	-----	set.computer.a	set.streamline.a	- font disk #1 -
type old english	- do not load -	set.computer.b	set.tech1.a	-copyright 1985-
edit unprotected	-----	set.d&d1.a	set.tech2.a	- comal users -
-----	pkg.rotate	set.fancy.b	set.thick chrs.a	- group,usa -
- comal fonts -	basic fonts	set.french.b	set.thin europ.a	-6041 monona dr-
-----	comal fonts	set.gothic.b	set.thin europ.b	- madison, wi -
font.mirror	-----	set.ital.nocur.b	set.type.b	- 53716 -
font.old english	- basic fonts -	set.italic.b	set.typeset.a	- 608-222-4432 -
font.pattern	-----	set.music note.b	set.typeset.b	-----
font.pet/graphic	-may be used by-	set.outline.b	set.underline.b	- may not be -
font.standard	- comal 0.14 -	set.repton.a	set.vic20.a	- copied or -
font.thin char	-program below -	set.rom fancy.b	set.vic20.b	- placed in -
font.vic20	-----	set.roman.a	set.warbot.a	- user group -
-----	set.80column.b	set.round.a	- comal 0.14 -	- libraries -
-comal 2.0 demo-	set.art deco.a	set.round.b	- font viewer -	-----



Typing Disk Front

hi	-----	+'- word list	52 Files	100 Blocks Free:
-----	home word list	misc word list	seq.keycodes	- comal users -
-comal programs-	+cr word list	+num word list	seq.typing	- group, usa -
-----	+fg word list	test word list	-----	- 6041 monona -
dvorak'keys	+l word list	-----	-comal package -	- madison, wi -
dvorak.14	+p word list	- articles -	src.dvorak	- 53716 -
typing test	+m word list	seq.dictionary	pkg.dvorak	-----
-----	+xb word list	seq.dvorak'hard	-for more info -	-(608)222-4432 -
- data files -	+wv word list	seq.dvorak'soft	- send sase to -	
-----	+qjk word list	seq.intro	-----	
- do not load -	+yz word list			



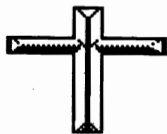
Typing Disk Back

hi	- do not load -	tuvwxyz	38 Files	216 Blocks Free:
-----		-----	seq.keycodes	- 6041 monona -
-comal program -	ab	- articles -	seq.typing	- madison, wi -
-----	cd	-----	-for more info -	- 53716 -
search dict	efgh	seq.dictionary	- send sase to -	-(608)222-4432 -
-----	ijklm	seq.dvorak'hard	-----	
- data files -	nopq	seq.dvorak'soft	- comal users -	
-----	rs	seq.intro	- group, usa -	





---HAVE---
---A---
---GOOD---
---DAY---



Library of Funcs & Procs

bootcomal.bas
c64 comal 0.14
comalerrors
hi
copy files.bas
control.l
link'program
accept.l
acos.l
arc'as.l
arc.l
arcl.l
arcr.l
ascii.l
asin.l
aspect.l
back'color.l
bin'str.l
bin.l
bit'ml.l
bitand.l
bitor.l
bitxor.l
border'color.l
bubsort'str.l
bubsort.l

cen.l
circle'as.l
circle.l
clearcoll.l
convert.l
cot.l
create.l
csc.l
curcol.l
currow.l
cursor.l
date.l
day'of'week.l
day'of'year.l
deg.l
delay.l
disk'get.l
draw.l
easter.l
eps.l
expb.l
fah.l
fp.l
get.l
getspritecolor.l
graphicstate.l

heading.l
hex'str.l
hex.l
inf.l
inkey.l
insertsort'str.l
insertsort.l
ip.l
jiffies.l
joystick.l
koala.l
leap'year.l
lightpen.l
load'screen.l
logb.l
lpad.l
ltrim.l
lwrc.l
matadd.l
match'r.l
matequal.l
matinvert.l
matmult.l
matscale.l
mattransp.l
matunit.l

142 Files

matvec.l
max.l
maxlen.l
metric.l
min.l
mount.l
move.l
obj'load.l
obj'save.l
paddle.l
page.l
pen'color.l
penstate.l
pi.l
play.l
plot'char.l
poly'as.l
poly.l
print'at.l
put'char.l
quicksort'str.l
quicksort.l
rad.l
randomize1.l
randomize2.l
rect.l

278 Blocks Free:

remain.l
replace'char.l
replace'str.l
reverse.l
rpad.l
rtrm.l
save'screen.l
screen'char.l
sec.l
select'lc.l
set'time.l
settime.l
setx.l
sety.l
shellsort'str.l
shellsort.l
shift'wait.l
showsprite.l
shuffle'str.l
shuffle.l
sid.l
sin2.l
spritestate.l
spritexcor.l
spritexsize.l
spriteycor.l

Library of Funcs & Procs - continued

spriteysize.l
squeeze.l
star.l

swap'str.l
swap.l
translate.l

turtle'size.l
turtlestate.l
uprc.l

val.l
xcor.l
ycor.l

COMAL Quick & Utilities 2

boot quick
load1
load2
0.14
-error-messages-
hi

1541'alignment
1541'align'1
1541'align'2

boot'dir'editor
comal'keypad.14
dir'manipulator
directory'editor
disk'edit/protct
disk'editor
display'seq'file
find
find'string/fast
find'string/full

ml'setup
names'printout
print'2'col'dir
remove comments
sd2 copier
sd2'copy&label
seq'to'speed
speed'to'seq
sprite'converter
sprite'editor

48 Files

text'dum'ctl-p
>--procedures--<
buffer.proc
cat'.proc
joystick.proc
load'obj.proc
loadshape.proc
ml'procs
paddle.proc
plot'char.proc

61 Blocks Free:

repeat'key.proc
restore'lbl.proc
saveshape.proc
tod.proc
wait'n'go.proc
>--data-files--<
dir'editor.mem
help.dat

Utilities 2 Back

> 1525 <
dump'1525
dumpscreen'1525
pretty'printer
> epson <
dual'epson'dump
fx-80'cmds.proc
> nec <
dump'nec8023a

nec'ml'dump
ml'dump.obj
nec'comal'dump
bigdump'nec.src
dir'print'nec
> cbm 8023 <
8023p'options
> okidata <
oki92'hi

oki92'screen'io
oki92.dump.obj
> gemini <
gem10x'lister
print'calendar
bit'map'print.l
> imp <
imp'dump
> prowriter <

44 Files

dump'prowriter
> 1520 <
1520/0.14demo1
1520/0.14demo2
1520/0.14demo3
1520/0.14demo4
1520/0.14demo5
1520/0.14demo6
1520/0.14demo7

185 Blocks Free:

1520/0.14demo8
1520/0.14demo9
1520/0.14demo10
1520/0.14demo11
1520/0.14demo12
1520/0.14demo13
1520/0.14demo14
1520'driver.proc

Captain C Gets Organized

boot c64 comal
c64 comal 0.14
comalerrors
generate errfile
hi
information84jun
help-comal
help-graphics
help-sprites
help-instruction

compare'dir
delete'dir

disk'summary
dos'menu
find'file
master'maker
print'dir
print'ids
startup
update
view'dir
>-----<
>procs follow- <
>-----<
choices.l

disk'get.l
dual'drive.l
file'exists.l
get'dir.l
init.l
intro.l
menu.l
menu2.l
menu3.l
page.l
print'dir'reg.l
print'dirlabel.l
printer.l

63 Files

quicksort.l
read'dir'part2.l
read'dir.l
read'dir2.l
screen.l
see.l
set'updated.l
sort'ids.l
type'of'dir.l
verified.l
>-----<
>end of comal <
> <

131 Blocks Free:

>the following <
>two programs <
>are written in<
> basic <
> ----- <
>do not load <
>them into <
>comal. <
>-----<
1541backup(free)
single file copy

Auto Run Demo Disk

boot c64 comal	clock	keno
c64 comal 0.14	clown	optical hexagon
comalerrors	color swirl	optical illusion
ml.sizzle	curves	optical triangle
hi	draw diamond	pie chart
menu	draw house	polyspirals
	example song	random music
- comal 0.14 -	graph equations	slide show
- programs -	graph waveforms	spirodotal
	graphics tutor	takeoff demo
arabesque	gutenberg	towers of hanoi
business graph	hilbert	
chris's star	ink blot	- data files -

63 Files

- below -

- do not load -

abc.sprite
alpha2.dat
blither.hrg
calculator.hrg
glady.hrg
griffin.hrg
rocket.sprite

-for more info -

1 Blocks Free:

- send sase to -

- comal users -
- group, usa -
- 6041 monona -
- drive -
- madison, wi -
- 53716 -

-(608)222-4432 -



Tutorial Disk

boot c64 comal	lesson four	lesson sixteen
c64 comal 0.14	lesson five	lesson seventeen
comalerrors	lesson six	lesson eighteen
ml.sizzle	lesson seven	lesson nineteen
hi	lesson eight	lesson twenty
menu	lesson nine	
	lesson ten	- special -
-comal lessons -	lesson eleven	- lessons -
	lesson twelve	
lesson one	lesson thirteen	graphics'tutor
lesson two	lesson fourteen	turtle'tutor
lesson three	lesson fifteen	

56 Files

- data files -
- below -

- do not load -

numberfile
studentfile

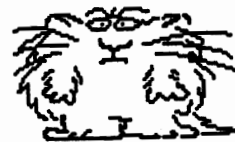
-for more info -
- send sase to -

- comal users -

7 Blocks Free:

- group, usa -
- 6041 monona -
- drive -
- madison, wi -
- 53716 -

-(608)222-4432 -



Bricks Tutorial Front

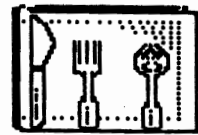
boot	edit1	headings
c64 comal 0.14	edit2	hello1
hi	fill	hello2
menu	frame	hello3
turtle'menu	getting started	hello4
cursor	getting started2	hello5
cursor2	graphics	hello6

35 Files

input1
input2
loops
loops2
move
plottext
size

3 Blocks Free:

steps1
steps2
steps3
taking control
turns
turtle
turtle2



Bricks Tutorial Back

boot	assignments	functions
c64 comal 0.14	case	graphics'review
hi	exam	if1
menu	files	if2
arrays	files2	loops
arrays2	files3	mod

27 Files

order
output
parameters
procedures
turtle'review
variables

35 Blocks Free:

---data files---
numberfile
studentfile



Games Disk 1

-----	eliza	-----
- comal 2.0 -	maze'game	battleship
- programs -	pigeons	bounce'ball
	puzzle'game	clue.14
boggle	santa'game	guess'it
breakout	wheel'of'fortune	hi-lo'game
clue	word'game	keno
comal'ace	yahtzee	magic'fruit
concentration		mastermind'word
docking	- comal 0.14 -	missing'letters
dog/cat	- programs -	sky'catcher

55 Files

slot'machine

- data files -
- do not load -

dat.game'names
dat.maze
hrg.field
missing.dat
slot'sprites

3 Blocks Free:

-games disk #1 -
- from -

- comal users -
- group, usa -
- 6041 monona -
- madison, wi -
- 53716 -

-(608)222-4432 -



Slide Show Disk 1

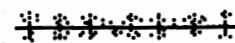
slide show	nudel.hrg	pattern1.hrg
directory	girl.hrg	linefig1.hrg
moire1.hrg	nude2.hrg	caveman.hrg
moire2.hrg	spacecraft.hrg	man.hrg
pattern2.hrg	earring.hrg	tetrad.hrg

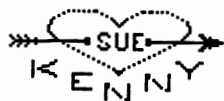
21 Files

snail.hrg
pattern3.hrg
petroom.hrg
linefig2.hrg
linefig4.hrg

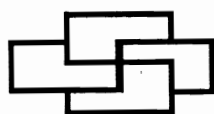
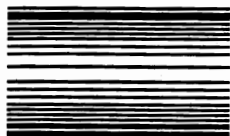
19 Blocks Free:

linefig3.hrg





Thank
you



COMAL Handbook Disk

boot c64 comal	>use enter <	>-----<	<u>144 Files</u>	<u>199 Blocks Free:</u>
c64 comal 0.14	>-----<	abs	exec	random
-----	create.l	and	exp	read
>-----<	cursor.l	append	false	ref
>error messages<	even.l	atn	file	repeat
> file <	fetch.l	case	for	restore
>-----<	file'exists.l	chr\$	func	return
comalerrors	get'char.l	close	if	rnd
>-----<	get'valid.l	closed	in	select
>file generator<	jiffies.l	cos	input	sgn
>-----<	lower'to'upper.l	data	int	sin
generate errfile	mount.l	delete	key\$	sqr
>-----<	new'employee.l	dim	len	step
>auto boot prog<	odd.l	div	let	tab
>-----<	pos.l	elif	log	tan
hi	put'record.l	else	mod	then
>-----<	randomize.l	end	not	to
>seq data files<	round.l	endcase	null	trap
>-----<	shift.l	endfor	of	true
information84mar	control.l	endfunc	open	until
help-comal	commodore'key.l	endif	or	using
help-graphics	take'in.l	endproc	ord	when
help-sprites	value.l	endwhile	otherwise	while
>-----<	>-----<	eod	pass	write
>comal handbook<	>sample progrms<	eof	print	zone
>procs & funcs <	>use load <	esc	printfile	>-----<
			proc	>bonus programs<

COMAL Handbook Disk - continued

>-----<	file'to'print	value/demo	quicksort/demo	disk'get/demo
print'directory	file'to'screen	shift/demo	joystick/demo	logical'ops/demo
utilities	cursor/demo	jiffy/demo	paddle/demo	

Structured Programming

boot c64 comal	example3.5	sec8.5b	<u>144 Files</u>	<u>158 Blocks Free:</u>
c64 comal 0.14	example3.6	sec10.2	example12.4	solution4.6
comalerrors	example3.6b	sec10.3	example12.5	solution4.7
hi	example4.2	sec10.4.1	solution3.4	solution4.8
information84mar	example4.3	sec10.4.1b	solution3.5	solution4.9
help-comal	example4.3b	sec10.4.1c	solution3.6	solution4.10
help-graphics	example4.5	sec10.4.2	solution3.7	solution4.11
help-sprites	sec4.4	quicksort.l	solution3.8	solution4.12
>-----<	example4.6	example11.1	solution3.9	solution4.13
sec1.5	sec4.5	example11.2	solution3.10	solution4.14
sec1.7	example4.7	example11.3	solution3.11	solution4.15
sec1.7b	example4.8	example11.4	solution3.12	solution4.16
probl.1	sec5.1	example11.5	solution3.13	solution4.17
probl.2	example5.1	example11.6	solution3.14	solution4.18
probl.3	sec5.2.2	sec12.3.1-write	solution3.15	solution4.19
probl.4	sec5.2.3	sec12.3.1-read	solution3.17	solution4.20
probl.5	sec5.2.3b	sec12.3.1b-write	solution3.18	solution4.21
sec2.5.1	example5.2	sec12.3.1b-read	solution3.19	solution5.1
probl.11	charcount.l	sec12.3.1c-read	solution3.20	solution5.2
probl.3.3	example8.1	setfile.l	solution3.21	solution5.3
cursor.l	example8.2	searchfile.l	solution3.22	solution5.4a
sec3.2	example8.3	validate.l	solution3.23	solution5.4b
sec3.2b	example8.3b	sec12.4-write	solution3.24	roulette.l
sec3.2c	fig8.1	sec12.4-read	solution3.25	solution8.1
example3.3	sec8.4.1	sec12.4-both	solution4.3	solution8.2
example3.4	sec8.5	example12.3	solution4.4	solution8.3
			solution4.5	solution8.4a

Structured Programming - continued

solution8.4b	solution8.7	solution8.10	solution8.13	>-----<
solution8.5	solution8.8	solution8.11	solution8.14	1541backup(free)
solution8.6	solution8.9	solution8.12	solution8.15	

PET COMAL 0.14

comal80/0.14	evaluator.l	recursions	<u>14 Files</u>	<u>419 Blocks Free:</u>
gencberrors.l	quicksort	hanoi	books & papers	info
cmbcomalerrors	utilities	formatter	bounce	remove //
			disk commands	

Packages Library Front

softscroll64	pkg.calchex	pkg.meta	<u>48 Files</u>	<u>350 Blocks Free:</u>
editor64	pkg.char	pkg.meta'rommed	lst.delink	- comal users -
startup	pkg.cmon'casbuf	pkg.ml	lst.list'package	- group, usa -
-----	pkg.cmon'rs232	pkg.oki92	-----	- 6041 monona -
-comal packages-	pkg.compactor	pkg.printer	- copies may -	- madison, wi -
- for c64 -	pkg.demo	pkg.text	- be made for -	- 53716 -
-----	pkg.exeq	-----	- personal use -	-----
pkg.basic	pkg.finchutilit	-comal programs-	- only -	-(608)222-4432 -
pkg.bitmap	pkg.first'last	-----	-copyright 1986-	-----
pkg.buffer	pkg.icon	labelmon	-----	-----

Packages Library Back

softscroll64	-----	src.calchex	<u>58 Files</u>	<u>126 Blocks Free:</u>
editor64	mac.16bitmath	src.cmon'casbuf	- source code -	-copyright 1986-
startup	mac.branching	src.cmon'rs232	-----	-----
-----	-----	src.compactor	-not commodore -	- comal users -
- comal symbol -	- source code -	src.demo	- compatible -	- group, usa -
- file for c64 -	-----	src.exeq	-----	- 6041 monona -
-----	- commodore -	src.first'last	merlin.finchutil	- madison, wi -
c64syms	- compatible -	src.mccomal	-----	- 53716 -
sym.comal64	-----	src.oki92	- copies may -	-----
syms	src.asmfix	src.printer	- be made for -	-(608)222-4432 -
-----	src.basic	src.text	- personal use -	-----
- macro files -	src.bitmap	-----	- only -	-----

COMAL 2.0 Packages Disk

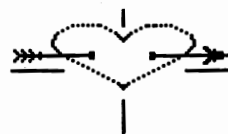
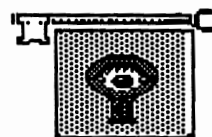
smon-comal	c64syms.pal	example.o	<u>20 Files</u>	<u>76 Blocks Free:</u>
c64syms	syms.mae	example.b	demo 1	symb alph
syms	cfname demo	errorpack.s	demo 2	symb num
c64syms.merlin	example.s	errorpack.o	proc.link'binary	print syms
			link'binary demo	show libraries

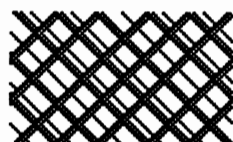
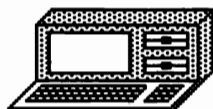
Cartridge Demo Disk 1

presentation	cos	exec	<u>141 Files</u>	<u>222 Blocks Free:</u>
--copy programs-	cursor	exit	mount	restore
single'drive'copy	data	exp	not	return
dual'drive'copy	delete	external	null	rnd
-external procs-	dim1	false	of	select
ext.add	dim2	file	open	sgn
ext.edit 40	dim3	find	or	sin
ext.sub	div	for	ord	spc\$
-----package----	do	func	otherwise	sqr
pkg.francais	elif	get\$	page	status\$
-handbook prg's-	else	handler	pass	step
abs	end	if	peek1	stop
and	endcase	import	peek2	str\$
append	endfor	in	pokel	tab
at	endfunc	input1	print	tan
atn	endif	input2	printfile	then
bitand	endloop	input3	proc	time
bitor	endproc	int	random	to
bitxor	endtrap	interrupt	randomize	trap
case	endwhile	key\$	read1	true
chain	eod	label	read2	until
chr\$	eof	len	read3	using1
close	err	let	read4	using2
closed	errfile	log	ref	val
con	errtext	loop	rename	when
copy	esc	mod	repeat	while
			report	writel

Cartridge Demo Disk 1 - continued

write2	---data files---	dat.random'read	dat.visitor'read
write3	dat.account'list	dat.test'loop	dat.winners
zone	dat.random'input	dat.visitor	





Cartridge Demo Disk 2

all'at'once
april'fool
arabesque2
bach'music
bat.commands
batch'copier
breakout
change'unit'#
create'fonts
curve1
curve2
dat.bwv779
dat.bwv781
dat.bwv794
dat.screens

differentiation
dragon
drawing-3d
error trapping
font.mirror
font.standard
formatted'list
func.binary\$
func.hex\$
get/set'screen
graph1
graph2
graph3
handler1
handler2

handler3
handler4
handler5
handler6
hilbert curves
lst.getnum
make'eprom'file
moire
moving'comal
moving'frank
paddles
playscore
queens
santa'game
shap.'a'

72 Files

shap.'c'
shap.'f'
shap.'k'
shap.'l'
shap.'m'
shap.'n'
shap.'o'
shap.'r'
shap.harry00
shap.harry01
shap.harry02
shap.harry03
shap.harry04
shap.harry05
shap.harry06

2 Blocks Free:

shap.harry07
shap.harry08
shap.harry09
shap.harry10
shap.harry11
shap.queen
shap.santa0
shap.santa1
shap.santa2
spiral'circles
spiralateral
tower'of'hanoi

Cartridge Demo Disk 3

all'at'once2
--demo programs-
1520 plotter
arabesque2
arabesque3
batchfile'editor
binary'counter
check'cartridge
curve3
curve4
extend'color

file'card'maker
graph4
graph5
graph6
koala'to'2.0
picture'loader
playscore2
protect64
read'directory
running'men
show'character

showlibs
sidmonitor
sound'envelope
sprite'editor
stampsprite
view'fonts
--batch files---
bat.commands
bat.font'cmds
---data files---
dat.bwv783

55 Files

dat.bwv786
dat.bwv801
---font files---
font.computer
font.d&d
font.greek
font.hebrew
font.rooski
font.standard
---pictures---
hrg.northwest

2 Blocks Free:

hrg.world'map
--procs & func-
func.modem'get
proc.show'sprite
---shape files---
shap.bat0
shap.bat1
shap.bat2
shap.men0
shap.men1
shap.men2

Cartridge Demo Disk 4

run'me'first
--demo programs-
all'at'once3
another'moire
auto'directory
bounce
cbm'to'comal
checkerboard
cones
copy'seq'file
crazy'quilt

datacollision
draw'flowers
draw'hourglass
draw'house
dual'drive'copy2
eight'boxes
factorial
ink'blot
logo'sampler
moving'boxes
moving'flag

mps801'dump
music'from'0.14
music'player
paint'circles
pascals'triangle
primes
print'time
quicksort
sierpinski
son'of'moire
spiral'squares

53 Files

spiral'stars
starwatch
three'pictures
---data files---
dat.bwv779
dat.bwv781
dat.bwv783
dat.bwv786
dat.bwv794
dat.bwv801
dat.instructions

16 Blocks Free:

--procs & func-
func.binary\$
proc.directory
proc.ellipse
proc.memorymap
proc.quicksort
proc.repeat'key
proc.stretch'x
proc.stretch'y

2.0 Tutorial Binder Disk

demoprogram
comal2.01
program 1
program 2
program 3
program 4
program 5
program 6
program 7
program 8
program 9
program 10
program 11
program 12

program 13
program 14
program 15
program 16
program 17
program 18
program 19
program 20
sprite 1
sprite 2
sprite 3
sprite 4
sprite 5
music demo

music 1
music 2
music 3
music 4
music 5
paddle game
joystick artist
lightpen demo
addr list demo
random file demo
move sequential
1520 plotter dem
train demo
thermometer

68 Files

batchfile'editor
bats
change unit
copy single 40
count
diamo
queen.spr
edit 40.ext
flowers
man
move comal
permute
playscore
primes

16 Blocks Free:

queens
quicksort
screen4
showlibs
sidmonitor
dat.bwv779
spriteeditor
c64symb
test.obj
test.src
skjul og hent.l
dat.bwv781

Slide Show Disk 2

slide show
directory
spiral.1.hrg
sue.hrg
sesame.st.hrg

snoopy.hrg
albert.hrg
dollar.hrg
dip.hrg
donald.duck.hrg

des.1.hrg
7-Shell.hrg
music.hrg
map.hrg
diane.hrg

21 Files

willy.hrg
raccoon.hrg
sincos1.hrg
watch.hrg
winston.hrg

19 Blocks Free:

micrometer.hrg

COMAL Sampler Disk

boot c64 comal	see'information	print'directory	<u>67 Files</u>	<u>106 Blocks Free:</u>
c64 comal 0.14	see'instructions	expression	shift/demo	-the following -
-error messages-	logo'book'sample	utilities	jiffy/demo	-two programs -
comalerrors	snowflake	recursions	quicksort/demo	-are written in-
-file generator-	sprite/turtle	formatter2	joystick/demo	- basic -
generate errfile	squiral	file'to'print	paddle/demo	- - - - -
-auto boot prog-	music	file'to'screen	disk'get/demo	-do not load -
hi	bounce	disk commands	logical'ops/demo	-them into -
---data files---	sprite'designer2	c64 comal info	--sprite images-	-comal. -
information84mar	lander	remove comments	lander'sprites	>-----<
help-comal	create'lander	see'roll/demo	sky'sprites	1541backup(free)
help-graphics	sky'falling	see'page/demo	-----	single file copy
help-sprites	create'sky	cursor/demo	-end of comal -	
-comal programs-	read'directory	value/demo	-demo programs -	

Graphics Primer Disk

boot c64 comal	demo 3.4	demo 2.4.1	<u>63 Files</u>	<u>278 Blocks Free:</u>
c64 comal 0.14	demo 4.1.a	demo 3.1.1	circle3.1	showsprite.1
comalerrors	demo 4.1.b	demo 3.2.1	getbackground.1	spritestat.1
hi	demo 4.2	demo 3.3.1	getborder.1	spritexcor.1
----programs----	demo 4.3	demo 3.4.1	getpen.1	spritexsize.1
menu	demo 4.4	demo 4.1.a.1	getpencolor.1	spriteycor.1
demo 2.1	flurry	demo 4.1.b.1	getspritecolor.1	spriteysize.1
demo 2.2	read'sprite.demo	demo 4.2.1	getturtlesize.1	turtlestat.1
demo 2.3	sprite'designer	demo 4.3.1	graphicstate.1	xcor.1
demo 2.4	--book listings-	demo 4.4.1	heading.1	ycor.1
demo 3.1	demo 2.1.1	---procedures---	hidescreen.1	--sprite files--
demo 3.2	demo 2.2.1	circle1.1	polygon.1	flake.dat
demo 3.3	demo 2.3.1	circle2.1	read'sprite.proc	
			showscreen.1	

C64 Graphics With COMAL

boot c64 comal	help-comal	background	<u>98 Files</u>	<u>232 Blocks Free:</u>
c64 comal 0.14	help-graphics	clear	turtlesize	red.1
-----	help-sprites	datacollision	sprite'aim.dat	cyan.1
>-----<	>-----<	drawto	sprite'datamak	purple.1
>error messages<	>comal programs<	fill	sprite'aim.1	green.1
> file <	>-----<	frame	rhianon.sprite	blue.1
>-----<	see'information	fullscreen	sprite'editor21	yellow.1
comalerrors	see'instructions	priority	define	orange.1
>-----<	-----	spriteback	sprite'box.1	brown.1
>file generator<	hidesprite	hideturtle	done.1	lt'red.1
>-----<	border	home	sprite'dot.1	dk'grey.1
generate errfile	setgraphic	identify	shape.1	med'grey.1
>-----<	settext	moveto	get'digit.1	lt'green.1
>auto boot prog<	forward	pencolor	box'filled.1	lt'blue.1
>-----<	left	plot	boxxy.1	lt'grey.1
hi	right	plottext	text'in.1	color'funcs.1
>-----<	pendown	spritecollisn	box.1	jiffies.1
>seq data files<	penup	spritecolor	pause.1	spriteback+
>-----<	setheading	spritepos	triangle.1	appendix'd
information84jun	back	spritesize	black.1	
			white.1	

Modem Disk

-----	-the following -	- features -	<u>68 Files</u>	<u>308 Blocks Free:</u>
- comal 0.14 -	- are public -	-----	midwestterm5.1	- xmodem -
-modem programs-	- domain -	-load the first-	term.c1	- features the -
-----	-programs which-	- file in each -	mlmid	- most common -
terminal	- load from -	- section with -	-----	-file transfer -
vt-52.v4	- basic -	- '8' and run -	- ravics term -	- protocol and -
-----	-----	- it -	-is a compiled-	-supports 1650 -
- comal 2.0 -	- they are -	-----	-basic program -	- type modems -
-modem program -	- provided to -	-----	-which features-	-----
-and procedure -	- allow use of -	- midwestterm -	- a buffer and -	xmodem-auto
-----	- certain file -	-is a compiled-	- automatic -	xmodem/autodial
proc.modem	- transfer -	-basic program -	- clock -	xmodem/ml
terminal'2.0	- protocol's -	-which features-	-----	
-----	- and other -	-a built in bba-	ravics term8.4	

Light Pen Demo Disk

lightpen'demo	colors'demo	proc.initpen	<u>7 Files</u>	<u>571 Blocks Free:</u>
func.color\$	pen'draw	proc.penkey	auto'dir-lpen	



LIGHT
MY FIRE



1



2.0 Auto Save

by Marvin Cook

How many times have you had to change the version number of the first line of your COMAL program? For example:

```
0010 // delete "programname.7"
0020 // save "programname.9"
```

Also, if you are chaining or using an external subroutine, the above commands do not work well because the calling program is looking for the exact name. The following COMAL 2.0 procedure will solve both problems.

```
0010 // scan
0020 // backup("save","autobackup")
0030 PROC backup(act$,f$) CLOSED
0040 USE system
0050 IF LEN(f$)>16 THEN f$:=f$(1:16)
0060 DIM f1$ OF 16
0070 f1$:=f$+".bk"
0080 IF LEN(f1$)=16 THEN
0090   f1$(14:16):=".bk"
0100 ENDIF
0110 DELETE f1$
0120 TRAP
0130   PASS "r0:"+f1$+"="+f$
0140 HANDLER
0150   NULL // if no backup file
0160 ENDTRAP
0170 PAGE
0180 PRINT AT 10,5: act$+"""0:";f$,"""
0190 PRINT AT 11,5: "CAT"
0200 CURSOR 10,1
0210 POKE 198,2 // causes an automatic
0220 POKE 631,13 // list and directory
0230 POKE 632,13 // listing
0240 ENDPROC backup
```

It can either be merged into the program you are developing or it can be an EXTERNAL procedure.

INSTRUCTIONS FOR USE

1. Set up the first two lines of your program as follows:

```
10 // scan
20 // backup("list","program name")
```

or

```
20 // backup("save","program name")
```

2. Merge the backup procedure into your program or copy it to your development disk and define in your program as an EXTERNAL procedure.

When you wish to save a backup copy of your developing program, erase the line number of line 10 and press return and then erase the line number of line 20 and press return. The backup process will erase the previous backup and rename the current file as "program name.bk" and save the latest version.

The only problem is that you have to have a version that does not have any structural errors or the scan command will produce an error. I find this a small price to pay.

Better yet, have the top two lines of your program set up the F8 key to do the SCAN and BACKUP for you. Just issue the commands (after the remarks) once, and after that the F8 key will do it all:

```
0010 // USE system
0020 // defkey(8,"147"scan"13"backup("
      save","name")"13") //wrap line
```

Now you have a "SAVE IT" key - the f8 key. A perfect match for your new POP key mentioned in this issue. ■

COMAL 2.0 Oki Data Graphics Dump

by Terry Ricketts

This is an all in COMAL graphic screen dump adapted from the Gemini 10X screen dump program in *COMAL Today* #9. The procedure will print the multicolor graphics screen on an Okidata 92 (or similar) printer using grey shading for the colors. Each bit pair representing one color dot is printed as a 2 dot vertical by 4 dot horizontal pattern. Since the dump is written in COMAL it is slower than the machine code dumps. Terry may provide us with a machine language version of this program in the future.

PROC dumpscreen CLOSED

```
//
// by Terry Ricketts
//
USE system
DIM a$ OF 404
OPEN FILE 4,"lp:/s8/l+",WRITE // open file for printer
PRINT FILE 4: ""24"" // reset the printer
PRINT FILE 4: ""27"%C075"" // move left margin
PRINT FILE 4: ""27"%9"8"" // set to 8/144 inch per line"
PRINT FILE 4: ""3"" // select graphics mode
PRINT FILE 4: ""3""10"" // clear the buffer
DIM cpat(0:15,2) // read color patterns
FOR i:=0 TO 15 DO
  FOR j:=1 TO 2 DO
    READ cpat(i,j)
  ENDFOR j
ENDFOR i
FOR col#:=0 TO 39 DO // do 40 cols
  FOR pair#:=0 TO 3 DO // 1 output row per pair of bytes
    a$:=""
    FOR row#:=24 TO 0 STEP -1 DO // do 25 rows per col
      add:=$e000+320*row#+8*col# // screen address
      loc:=$d800+40*row#+col# // color ram address
      setpage(6) // select color ram
      coloram:=PEEK(loc) MOD 16 // get color screen info
      backx:=PEEK(53281) MOD 16
      setpage(0) // select ram under rom
      scrmem:=PEEK(loc)
      scrmemhi:=scrmem DIV 16
      scrmemlo:=scrmem MOD 16
      FOR byte#:=7 TO 0 STEP -1 DO //read 8 bytes per char
        valx:=PEEK(add+byte#) // read the screen
        CASE pair# OF // get the bit pair for each color
          WHEN 0
            valx:=valx DIV 64
          WHEN 1
            valx:=(valx DIV 16) MOD 4
          WHEN 2
            valx:=(valx MOD 16) DIV 4
```

```
          WHEN 3
            valx:=valx MOD 4
          ENDCASE
        CASE valx OF // figure out color of pair
          WHEN 0
            valx:=backx
          WHEN 1
            valx:=scrmemhi
          WHEN 2
            valx:=scrmemlo
          WHEN 3
            valx:=coloram
          OTHERWISE
            CLOSE
            STOP "err in case"
          ENDCASE
        a$:=a$+CHR$(cpat(valx,2))+CHR$(cpat(valx,1))
      ENDFOR byte#
    ENDFOR row#
    a$:=a$+"3"+"10"" // add lf to line
    PRINT FILE 4: a$ // send 1 line to printer
  ENDFOR pair#
ENDFOR col#
PRINT FILE 4: ""3""2""
CLOSE
DATA $0f,$0f,0,0,$0f,0,5,$0a // color codes for pairs of bits
DATA $0a,$0a,6,9,$0d,$0b,8,2
DATA $0a,1,7,$0e,5,0,$0d,$0e
DATA 6,7,4,2,1,8,0,9
ENDPROC dumpscreen
```

Comal is a
nice language.
You can buy
alot of stuff.
If you buy the
playnet disk you
can talk to
people.

By Rhianon
age 8



Graphics Editor System

by Colin Thompson

Overview

The Graphics Editor is a system of 26 COMAL 0.14 and M/L programs, all interlinked through a series of menus. The primary purpose of the system is to manage a library of bitmap pictures. The system includes programs that convert source pictures to COMAL format, repair pictures, display the pictures on the screen, print the images to any printer, add lettering to a picture, and change the bitmap format to a compacted form to conserve space on the disk.

Programs Included in the System

The system is distributed on the COMAL 0.14 side of COMAL Today Disk #11. The disk includes a fastloader. The HI program is the only one you may LOAD or CHAIN. You may not directly LOAD or CHAIN any other program on the disk. The following files make up the Graphics Editor System.

HI - Expands memory and CHAINs the next file. This is the only entry point into the system!

FILE.1.J - Main system menu. It loads most of the machine language.

FILE.2.J - Slide Show. A bitmap viewer.

FILE.3.J - Editor. Edit, LOAD, or SAVE bitmap pictures.

FILE.4.J - Print Menu. Select your printer and interface, then print the picture.

FILE.5.J - Disk Directory Printer.

FILE.6.J - Utilities. Edit or create the DIRECTORY file. Disk manager. Instructions reader/printer.

FILE.7.J - This text file.

FILE.8.J - Gutenberg Lettering Press. Add lettering to pictures.

FILE.9.J - Gutenberg data file.

FILE.10.J - Machine Language file used by the Editor for mirror images.

FILE.11.J - Source file Converter. Converts BASIC bitmaps, Doodle, and COMAL Hires files to COMAL Bitmaps.

FILE.12.J - Compact System Menu.

FILE.13.J - Compact Filewriter. Updates the COMPACT PIX and DIRECTORY files.

FILE.14.J - Compactor. Converts COMAL Bitmaps to Compacted form.

FILE.15.J - Compact Viewer. Compact picture viewer. Multi-purpose.

FILE.16.J - Uncompactor. Converts Compacted Bitmaps to COMAL Bitmaps.

DIRECTORY - Sequential file that holds the filenames of the COMAL Bitmaps on the disk.

COMPACT PIX - Sequential file that holds the filenames of all the Compacted Bitmaps on the disk.

Screen Dump Programs

DUMP.1520 - Commodore's 1520 plotter.

DUMP.1525 - 1525, 801 and compatibles.

More ►

[illegible]

DUMP.BX80 - Panasonic BX80 and other Epson compatibles.

DUMP.EPSON - Epson, Gemini and compatibles.

DUMP.IMP - Imp printer.

**DUMP.NEC - NEC 8023 and C.Itoh
Prowriter.**

DUMP.NEC.B - Double size dump.

DUMP.OKI92 - Okidata 92.

DUMP.OLIV - Olivetti ink jet.

You will also find two COMAL Bitmaps and two Compact Bitmaps on the disk.

Hardware Requirements

Disk drive: 1541, 1571, MSD, or compatible drive. If you use a dual drive, you may keep the Graphics Editor disk in drive 1 and a picture disk in drive 0. All pictures are LOAded from and SAVED to drive 0.

Joystick: Any high quality joystick or trackball. Kraft's short handle \$12 joystick is recommended. Wico's top line trackball is also good.

File Copier: Any good Commercial copier like Fast Hackem, or Public Domain copier will do. If you have a single drive, you will use your copier a lot, so get a good one.

Optional Equipment

COMAL 0.14 cartridge. Available from Peripherals Plus. See *COMAL Today* #6 page 45.

Printers. The system supports most commercial printers. If your printer and interface can be made to be 1525/801 compatible, you will experience no printing problems. If not, you may have to adjust your interface to be transparent.

David's Directory Designer. This \$15 BASIC program is indispensable for organizing disk directories. See *COMAL Today* #9, page 63.

COMAL 2.0 Cartridge. Just because.

Operating Instructions

Before you do anything, make a backup copy of the master disk.

Use your disk copier. The disk is not protected. Do not attempt to use the original disk. You cannot WRITE to it. After you make a backup disk, make a work disk. The work disk will not contain the COMAL language files. From your backup disk, copy all of these files onto your blank work disk:

"hi"

" file.1-16.j" (All 16 System files)

" directory"

" compact pix"

Copy the "dump." file that matches your printer.

Also copy any Bitmaps or Compacted pictures that you want to use.

This will be the disk you use.

Next, format two blank disks to hold picture files. From COMAL, do this:

More ►

[illegible]

Insert the 1st disk.

```
pass "n0:bitmap pictures 1,mn"
list 10 " directory"//one leading space
```

Insert the 2nd disk.

```
pass "n0:compact pix 1,bv"
list 10 " compact pix"//1 leading space
```

These will be the first disks in your library. You can record up to 20 bitmaps on a Bitmap Disk, and up to 60 Compacted pictures on the Compact Pix Disk.

A Tour Through the System

Your first stop will be at the Main System Menu. This is a central place in the in system. From here you may travel to four places:

Editor
Gutenberg
Slide Show
Utility Menu

The words "SELECT COMMAND:", shimmering in blue, followed by the flashing red letters "EGSU", prompt you to press one of these four keys. Most of the menus in the Bitmap section are a variation on this theme.

Press S to go to the Slide Show program. You will be asked to insert a "*Slide Show Disk*". This program requires a disk with Bitmaps and an accurate "directory" file. Use this program to look at the bitmaps you've collected. After you've looked at the bitmaps on your work disk, press M for Main Menu.

From the Main Menu, select E for Editor. Plug in your joystick to port 2 and wait for the Editor's Menu to appear.

This is a very complex menu, divided into two parts. The left third of the

screen has a list of menu options, each in a different color. The right two thirds of the screen show the commands you may invoke when using the editor. The command line at the bottom of the screen shows the commands available now:

SELECT COMMAND:

[left arrow]*VDSL PUGC[british pound]

Command explanation:

[left arrow] toggles between this screen and the high res graphics screen.

[*] is the entry into the Compacted Picture System.

[V]iew "directory" file. Reads the file and puts the first filename on the command line. You may then [L]oad that bitmap, see the [N]ext filename, or [Q]uit looking.

[D]isk Catalog. Press return to see the disk's catalog on the screen. Or you may enter [P] to print the catalog on your printer.

[S]ave the bitmap currently on the graphics screen. You may change the filename or press return on the command line to accept the filename presented. Pictures may be SAVED repeatedly. If you change your mind, press the space bar once, erasing the first letter of the filename, then press return.

[L]oad a bitmap. Enter the filename and press return. You may abort by erasing the first letter. You never need to include the ".hrg" filename suffix that all bitmap pictures carry.

[P]rinter Menu. Select this to print the current picture on your printer. The

More ►

Editor will appear after printing stops.

[U]tility Menu. For details, see the explanation later.

[G]utenberg. This is the lettering program.

[C]onvert source files to COMAL Bitmaps.

[british pound] calls the Slide Show program.

Editing Mode Commands

When you press the left arrow key (not the cursor left key), the high res screen is shown. The following options are now active. Just press the corresponding key.

- A large multicolored X is the cursor. Move the cursor with the joystick. Bitmap pictures are black and white. To change the color of any pixel, move the cursor over that pixel, select the color to "paint" with the *f7* key, and press the fire button. The border color reflects the current drawing color. You can draw a line by holding down the fire button and moving the cursor.

The cursor speed may be varied. *F1* toggles between fast (8) and slow (1). The speed may be varied by one with the [-] and [+] keys.

[H] homes the cursor. The cursor
up/down/left/right keys move the cursor
to the border in that direction. The
SPACE BAR hides the cursor.

F4 erases the screen. **[R]** reverses the screen. **[M]** makes a mirror image. **[F]** fills from the cursor position in the selected color.

[L] draws a straight line between two points. Pressing [L] sets the beginning point. Move the cursor to the end point and press fire to draw the line.

[D] puts the pen down until the fire button is pressed. Text may be plotted on character block boundaries with [T]. [W], window is like [L], but draws a rectangle around two points. A shaded pattern results with [S].

The following commands draw various geometric designs. [B] draws a series of boxes or rectangles vertically or horizontally. You supply the size, number and direction.

[O] draws a tall oval on the screen. This oval will print on paper as a circle, due to differing aspect ratios. A true circle can be drawn with the [P] option.

[P] (polygon) draws a figure with any number of sides. A 50 sided figure looks like a circle on the screen, but will print on paper as a wide oval.

See the bitmap printed on page 72 of *COMAL Today* #9 for examples of P,B,S,W,F,T, and O.

Helpful hints for using the Editor.

There is no "oops" key. If you make a serious mistake, you will have to re-load the bitmap. For this reason, you should frequently SAVE your work.

The Editor is ideal for repairing or modifying existing bitmaps. It was not designed to draw original art. Doodle! is much better at that, and Doodle! pictures may be converted to COMAL

More ►

Question

General Notes on the Entire System

Any program can be stopped with the STOP key, and then continued with RUN. Pictures will stay on the screen until you erase them. The Bitmap section remembers the current filename. The Compact System does not.

This system is a treasure trove of advanced COMAL 0.14 programming techniques, including many procedures and functions you may wish to pull out and store on disk for future use.

As you work with the system, you will be shuffling three disks: the work disk, a picture disk, and source file disk. Label them clearly. If you insert the wrong disk, the system will not crash.

Do not attempt to make programming changes until you have mastered the method used to pass parameters between modules. Location 750-760 hold these values.

This is largest COMAL 0.14 program ever written. When in doubt, follow the instructions on the screen.

The entire Graphics Editor System is copyrighted. All rights reserved by the author. You may legally copy this disk and give it to your friends, but no one may sell the System or any part of it, except the COMAL Users Group, USA, Ltd.

Special Note: there is one leading space in many of the file names used in this system. This is intentional and it will not work without the space. Each " dump", and " file" file plus " directory" and " compact pix".

IBM PC COMAL

Greetings- I could use some information on IBM-PC COMAL. I am writing a program to control an X-Y table for my neighbor who builds lasers as a part-time business. The program is mostly done, enough that he can demonstrate some possible applications, but he ran into a business "snob factor".

It seems one potential customer liked what he saw, but did not want a "toy" computer running his laser. My neighbor and I discussed it a length and came to the sorry conclusion that the control program will have to be moved to an IBM-PC for some customers. Or they simply will not buy.

Questions: How well does the PC COMAL support graphics? Does it offer packages, such as for joystick control and turtle graphics? These are integral to the C-64 program and I would want them to run on the IBM-PC as well. I have a Compaq portable that I could use for the development and testing. Whatever info you have would be appreciated. Thanks for the assistance.

- Dennis O Johnson, Wadena, MN

Answer: Yes IBM PC COMAL supports graphics (both x/y and turtle), but I believe there are some bugs in that package. It also accepts linking packages to programs just like the C64 COMAL 2.0 Cartridge. Graphics, System, and Memory packages come with the system. I am not sure about joystick control as I don't know how it is treated by the IBM PC. I have used IBM PC COMAL on a Compaq portable myself, so you can be sure that it will run. Your program should transfer from your C64 to the IBM PC just fine. ■

Text Package

by Dick Klingens

When we created a large monitor program with a lot strings in which we stored a help menu, we were not able to extend that program with more disk operations, because all memory was occupied.

Two possibilities were left; leaving out the help menu or storing the help strings in another part of the memory.

We did the latter. We created a RAM disk (a text buffer) as a package and we called that package TEXT.

The new package has 4 procedures and one function. You will notice that the commands are similar to the file commands of Pascal and work in much the same way.

```
PROC readln(REF x$)
PROC writeln(REF x$)
PROC reset
PROC rewrite
FUNC eot
```

READLN fetches a string from the buffer. During this fetch there is a test on reading the end of the buffer. If so, an error message is printed and the program is stopped.

WRITELN does the reverse. It writes a string into the buffer. If the 16K buffer is full, the message 'out of memory' is printed.

RESET directs the reading pointer to the first position in the buffer. This statement can be used to read again from the beginning.

REWRITE directs the reading and writing pointer to the first buffer position. It

empties the buffer!

EOT (End Of Text) is a function that returns TRUE when the reading pointer is in the same position as the writing pointer. If EOT=TRUE, then there is no more text in the buffer. This function is similar to the EOD function built into COMAL.

The following example shows how to use this package.

```
USE text
rewrite // empties buffer
DIM x$ OF 40
PRINT "Enter any text. Press RETURN on"
PRINT "a blank line to end."
REPEAT
  INPUT x$
  IF x$<>"" THEN writeln(x$)
UNTIL x$=""
reset // read pointer to first position
WHILE NOT eot DO
  readln(x$)
  PRINT x$
ENDWHILE
END "All done."
```

This package is valuable to programmers who need access to lots of text without using the disk drive. One use might be in a bulletin board program to speed up menu printing.

The source code for this package is too long to list, so it and the assembled package are on *Today Disk #11*. **Special note:** The *DEMO/TEXT2* program also on the disk shows that any text in the RAM disk buffer is also saved with the program.

More ►

Text Package - continued

```

; src.text (comal module)
; by m.bokhorst, nov85
; revised by d.klingens
; dutch comal users group
;
;- variables & constants -
defpag = %01000110
dummy = $ca2f
proc = 112
endprc = 126
func = 227
endfnc = 126
pshint = $c9ce
str = 2
ref = 117
point = $fb
fndpar = $c896
copy1 = $45
copy2 = $47
copy3 = $49
copydn = $c8a2
runerr = $c9fb
;
;-- module --
* = $8009
.einde .word end
.word dummy
.byte 4,'text'
.word procs
.word reset
.byte 0
;
;- procedures
;& functions --
procs .byte 7,'rewrite'
.word hempty
.byte 7,'writeln'
.word hput
.byte 6,'readln'
.word hget
.byte 5,'reset'
.word hres
.byte 3,'eot'
.word heat
.byte 0
;
;- headers --
;
hempty .byte proc
.word empty
.byte 0
.byte endprc
;
hput .byte proc
.word put
.byte 1
.byte str+ref
.byte endprc
;
hget .byte proc
.word get

```

```

.byte 1
.byte str+ref
.byte endprc
;
hres .byte proc
.word reset
.byte 0
.byte endprc
;
heat .byte func
.word eot
.byte 0
.byte endfnc
;
;-- code --
;
empty lda #<end
ldy #>end
sta einde
sty einde+1
;
reset lda #<end
ldy #>end
sta point
sty point+1
rts
;
eot jsr teof
lda #0
roi a
tax
lda #0
jmp pshint
;
put lda #1
jsr fndpar
lda copy1
clc
adc #<2
sta copy1
lda copy1+1
adc #>2
sta copy1+1
lda einde
ldy einde+1
sta copy2
sty copy2+1
ldy #1
setup lda (copy1),y
sta copy3,y
dey
bpl setup
jsr len
lda point
clc
adc copy3+1
sta point
lda point+1
adc copy3
sta point+1
jmp copy
noroom lda (copy1),y
sta copy3
pha
iny
lda (copy1),y
sta copy3+1
pha
lda point
ldy point+1
sta copy1
sty copy1+1
jsr len

```

```

jmp copydn
;
eof ldx #201
.byte $2c ;skip 2
out ldx #52
jmp runerr
;
teof lda point
sec
sbc einde
lda point+1
sbc einde+1
rts
;
get jsr teof
bcs eof
lda #1
jsr fndpar
lda copy1
clc
adc #<2
sta copy2
lda copy1+1
adc #>2
sta copy2+1
ldy #1
lda (copy1),y
sec
sbc (point),y
dey
lda (copy1),y
sbc (point),y
bcc noroom
lda point
ldy point+1
sta copy1
sty copy1+1
ldy #1
setup1 lda (point),y
sta copy3,y
dey
bpl setup1
jsr len
lda point
clc
adc copy3+1
sta point
lda point+1
adc copy3
sta point+1
jmp copy
noroom lda (copy1),y
sta copy3
pha
iny
lda (copy1),y
sta copy3+1
pha
lda point
ldy point+1
sta copy1
sty copy1+1
jsr len

```

```

ldy #1
lda (point),y
clc
adc point
tax
dey
lda (point),y
adc point+1
tay
txa
clc
adc #<2
sta point
tya
adc #>2
sta point+1
ldy #1
pla
sta (copy1),y
dey
pla
sta (copy1),y
copy ldx copy3
lda copy3+1
tay
beq l001
eor #255
tay
iny
clc
lda copy1
adc copy3+1
sta copy1
bcs l002
dec copy1+1
l002 clc
lda copy2
adc copy3+1
sta copy2
bcs l003
dec copy2+1
l003 lda (copy1),y
sta (copy2),y
iny
bne l003
inc copy1+1
inc copy2+1
l001 dex
bpl l003
rts
;
len lda copy3+1
clc
adc #<2
sta copy3+1
lda copy3
adc #>2
sta copy3
rts
end .end

```

De-LINK a Package

by Dick Klingens

The new version of "Rod the Roadman" by Borge Christensen has a nice linked package: doppelskaerm (danish for dual screen). The package copies (danish: gem) a hires screen into another memory area. And after storing a screen it is possible to swap (danish: skift) that screen with the current hires screen.

I wanted to use that package in another program, but I first had to separate the package from the program. DeLINKing a package from a program is easy to do.

Load the program with the linked package and type:

DEL 1-

This removes all lines of the program leaving only the package in memory.

Then bring into memory the program printed below (also on *Today Disk #11*) by typing in the program or using the command (don't use LOAD):

MERGE "lst.delink"

If you had LOADED this program, or did a NEW to remove the old program, all non-ROMMED packages would be erased from memory. Once in memory, simply RUN the program and it will write a linkable package file with the name you specify. The package can then be linked onto other programs.

```
// DELETE "lst.delink"
// LIST "lst.delink"
// by Dick Klingens - nov85
// Dutch COMAL Users Group
//
USE system
info
DIM filename$ OF 18
liblo:=&c7f0; libhi:=&c7fa
libpag:=&c804; libpt:=&c7ef
nlib:=PEEK(libpt)
show'names; choice(num)
lo:=PEEK(liblo+num-1)
hi:=PEEK(libhi+num-1)
pag:=PEEK(libpag+num-1)
setpage(pag); ad'start:=lo+256*hi
ad'end:=PEEK(ad'start+1)
ad'end:=+256*PEEK(ad'start+2)
get'filename
mem'to'obj(1,ad'start,ad'end-1)
END " Done"
//
PROC show'names
PAGE
PRINT " DELINK"
PRINT
PRINT " Packages in memory are:"
PRINT
FOR t:=1 TO nlib DO
  lo:=PEEK(liblo+t-1)
  hi:=PEEK(libhi+t-1)
  start:=lo+256*hi
  pag:=PEEK(libpag+t-1)
  setpage(pag)
  PRINT USING " ##) ": t,
  PRINT name$(start)
ENDFOR t
ENDPROC show'names
//
FUNC name$(x) CLOSED
l:=PEEK(x+5)
// length of package name
DIM r$ OF l
FOR t:=1 TO l DO
  r$:=CHR$(PEEK(x+5+t))
ENDFOR t
RETURN r$
ENDFUNC name$
//
PROC choice(REF num) CLOSED
IMPORT nlib,currow,curcol
x:=currow; y:=curcol
REPEAT
  INPUT AT x,y,2: " Type number: ": num
  UNTIL num>=0 AND num<=nlib
  IF num=0 THEN
    END "End of program"
  ENDIF
ENDPROC choice
//
PROC info
PAGE
PRINT " DELINK"
```

More ►

Differentiation

```

PRINT
PRINT " This program deLINKs a package that"
PRINT " is linked to another program."
PRINT
PRINT " First LOAD that program and type:"
PRINT "  DEL-"
PRINT "  MERGE ""lst.delink""
PRINT "  RUN"
PRINT
PRINT " The program will create a LINKable"
PRINT " package file on disk."
PRINT
PRINT " *** Type any key to continue or  ***"
PRINT " ***          ESC to stop  ***"
WHILE KEY$<>"0" DO NULL
WHILE KEY$=""0" DO NULL
ENDPROC info
//
PROC mem'to'obj(f'num,add,last) CLOSED
DIM code$ OF 80, adrs$ OF 4
lino:=0
REPEAT
  lino:+1; rl:=0; code$:=""
  WHILE rl<$18 AND add+rl<=last DO
    code$:+hex$(PEEK(add+rl)); rl:+1
  ENDWHILE
  adrs$:=hex$(add DIV 256)+hex$(add MOD 256)
  code$:=";" +hex$(rl)+adrs$+code$
  checksum(code$)
  PRINT FILE f'num: code$
  add:+rl
UNTIL add>last
code$:=";00"+hex$(lino DIV 256)
code$:+hex$(lino MOD 256)
code$:+code$(4:)
PRINT FILE f'num: code$
CLOSE FILE f'num
//
PROC checksum(REF x$)
ch:=0
FOR t:=2 TO LEN(x$) STEP 2 DO
  ch:+VAL("$"+x$(t:t+1))
ENDFOR t
x$:+hex$(ch DIV 256)+hex$(ch MOD 256)
ENDPROC checksum
//
FUNC hex$(x) CLOSED
DIM h$ OF 16
h$:="0123456789abcde"
RETURN h$((x DIV 16)+1)+h$((x MOD 16)+1)
ENDFUNC hex$
ENDPROC mem'to'obj
//
PROC get'filename
INPUT AT 0,0,18: "Enter filename: ": filename$
TRAP
  OPEN FILE 1,filename$,WRITE
HANDLER
  CLOSE
  END "Disk error --";ERRTEXT$
ENDTRAP
ENDPROC get'filename

```

by Tom Kuiper

Cartridge Demo Disk #2 has a program which, for some readers, is by itself reason enough to buy the Cartridge. (You mean you don't have one yet?) The program is called *Differentiation*. It does symbolic differentiation of functions coded in "computerese". For example, the derivative of $\sin(x)$ is $\cos(x)$. For that you don't need a computer. But how about the derivative of $\exp(-4*\ln(2)*((x-a)/w)^2)$ with respect to w ? Indeed, that would take me 20 minutes of algebraic scribbling to get it wrong!

I've modified the original program to facilitate interactive use. College freshmen should not use it to do their calculus homework. (You might have to do a derivative sometime without your Commodore. Learn how!) The program comes up with an instruction screen. After you hit any key, you will be asked to input a function. Use only single letter lower case variables. Next you are asked for the variable with respect to which you want to differentiate. The result is displayed shortly. The cursor is positioned back at the variable name input. You can specify the variable with which the **RESULT** is to be differentiated. You can repeat this until the result string overflows. If you respond with a blank, then the cursor goes back up for another function input. You might redo your original function with respect to another variable, edit your original function, or type in a completely new one. If you enter a blank line the program ends.

[While previously printed in COMAL Today #9, the program was left off its disk. The program is on Today Disk #11] ■

FFT

by Tom Kuiper

In *COMAL Today* #7 Professor Olson of the University of Minnesota mentioned an interest in a Fast Fourier Transform (FFT, for short) program. As it happens, I coded one up about a year ago to test some ideas about the behaviour of a hardware FFT that I use for spectrum analysis. Generally, when I contribute something technical, I like to include a simple explanation (see "Interstellar Dust Clouds" in *COMAL Today* #7). However, saying something simple about the FFT was a severe challenge. The alternative was to show what an FFT does, without trying to explain it. This required writing a demo program. In the end, I attempted to both, with this result:

Jean Baptiste Joseph de Fourier lived from 1768 to 1830. Among his many accomplishments, he showed that almost any functions' function could be represented by an infinite series of sines and cosines at frequencies which are 0, 1, 2, 3, 4, 5, times some fundamental frequency. The "spectrum" of a signal is a measure of the amount which each of these sine/cosine pairs contributes to the total signal. Since most signals are limited at the upper frequency end (for instance, those coming from your HI-FI speakers), we don't actually need an infinite series. In most practical applications, we can use a finite series. The process of finding the spectrum from a signal is straight forward, but tedious. To get the coefficient (multiplier) for a given sine (or cosine) in the series, you multiply the signal by that sine (or cosine) for each time step, and then sum all the resulting values. If the signal consists of N samples, this requires $N*N$

multiplications for the sines and $N*N$ more for the cosines. We say that the computation size is 'of order N squared'.

In April of 1965, J. W. Cooley and J. W. Tukey published "An algorithm for machine calculation of complex Fourier series" in the journal *Mathematical Computation*. The size of the computation is of order $N*\log(N)$ instead of $N*N$, a considerable saving when N is large! An interesting history of how Tukey's method came to be published is found at the end of chapter 1 of Brigham's book *The Fast Fourier Transform*. It turns out that the method had been known as early as the turn of the century. A completely analog method, using hybrids junctions and coaxial cable delays, was devised by Butler (independently, as far as I know) for phased antenna arrays.

The FFT algorithm presented here is adapted from the FORTRAN program given by Brigham (Fig. 10-7). To illustrate it, the demo program *FFTTEST* uses the C64 sound chip to provide test signals. The signal is then plotted in the lower half of the screen. Since this is a real --not complex-- signal, I use the algorithm given in Fig. 10-10 of Brigham to make more efficient use of the basic complex FFT. Because the resultant transform is complex, I form the power spectrum by squaring and adding the real (cosine) and imaginary (sine) components. The power spectrum is then plotted in the upper half of the screen.

By trying out different types of waveforms, you may get some idea of why they sound different. It is particularly interesting to try the pulse waveform with different duty cycles, and to discover why a duty cycle near 50% gives

More ►

5. A set of procedures to facilitate authoring interactive discussion type lessons (or jokes for that matter.)

9. A typing tutor (One has already been announced in *COMAL Today*, but here are my design criteria anyway):

More ►

Educator Needs

by Jim Ventola

Most readers of *COMAL Today* already know that COMAL is the optimal educational computing language. But as a teacher in an English department at a community college, I see computers not only as objects of study in themselves but as tools for writing. So while I agree with those who say COMAL is the language schools should use to introduce students to computing, I also see COMAL as a tool for teachers interested in fields other than computing, math or science.

An English teacher is likely to put "wordprocessing" at the head of the list of useful things a computer can do. Learning a language like COMAL would come much later, if at all. But once the plunge into programming is taken, COMAL is the best choice there as well. For example, many teachers have spent time learning PILOT because it promises a way to write interactive lessons without having to learn a computer language. Actually, PILOT is not much easier to learn than COMAL and provides no way to write applications other than interactive lessons. Indeed, even what it does do it must do less powerfully than COMAL could since PILOT is a template to protect the user from the machine rather than a language to give him or her access to it.

In any case, teachers learning PILOT are attempting to write *applications* of their computers. So, it is important to think of applications as well as of the formal features of COMAL as a computing language when we think about COMAL in education.

Until COMAL is well implemented on Apple computers, it cannot gain its rightful

place in the curriculum of American schools. Meantime, it is gaining popularity and provides its home users with a powerful educational experience and tool. If there were more useful applications in COMAL for teachers to see, I think many of them would follow up and learn the language. For example, I am in the midst of assembling a library of routines to closely (but not slavishly) emulate PILOT for writing interactive dialogs. The music, graphics, and file handling of COMAL are vastly superior to PILOT's and there is a huge amount of COMAL applications code that I can study and adapt. (So look for FrameWriter-2.01 one of these years.) The corpus of COMAL programs, what could be called the "culture of COMAL,"--but really more the "community of COMAL," given its open, un-protected style--is one of COMAL's great strengths.

There are other applications of COMAL that I would like to see. But most of them are beyond my abilities as a programmer. That is one of the frustrations of using COMAL. It's so easy you quickly get so you can imagine writing programs you wouldn't dare dream of in BASIC, but still find beyond you. One solution is to pool talents. So I will describe my wished-for programs more fully, in hopes that someone else will decide to write them.

1. An outline processor.
2. A series of machine language routines that could be made into packages for use by COMAL programmers. One set, for example, might emulate the control keys of the COMAL editor within any program. Then, if most COMAL programs used these routines, there would be a "common interface" among COMAL applications

More ►

Questions and Answers

K Keller of Ladora, Iowa has a lot of questions including:

SUBMITTING MATERIAL

Question: What's the story on submitting stuff to the COMAL Users Group USA Limited? I converted some simple BASIC stuff to COMAL and would like to send it to you.

Answer: We welcome user submissions of programs and articles. Send them on disk with the text in either PaperClip, EasyScript, WordPro, Paperback Writer, or SEQ type files. If we use your material, we will send you one of our User Group disks in return (you can specify which one you are interested in).

ON ... GOTO --> CASE

Question: How is BASIC's ON .. GOTO or ON .. GOSUB handled and written in COMAL?

Answer: COMAL's CASE structure handles the various case's referred to in the ON .. GOTO or ON .. GOSUB code in BASIC.

ASC and ORD

Question: What does ORD stand for as the COMAL equivalent of BASIC's ASC?

Answer: ORD stands for ordinal, while ASC stands for ASCII. They both mean the numeric value assigned to the character.

BEST OF DISK TIPS

Question: On the *Best of COMAL 0.14* disk, how does one really use "DISK'DATA'BASE"? I'd like it to keep track of commercial software place

addresses and personal collections. How does one copy "dbase" relative/random file on the flip side of this disk? On the same disk, "guess'it" is a dandy game. What's the code to look at that enables the program to write its own DATA for use in other programs? What changes need to be made to the game so that it guesses any noun? And when it is first run I want it to ask, "Is it alive"? What needs to be done so that it does that?

Answer: The disk data base you are referring to is discussed on page 50 of COMAL Today #6. It is a very complex system and cannot easily be changed or modified. You may be more interested in the data base programs published on Today Disk #8 and explained on page 25 of COMAL Today #8. Guess'it is another complex program and is discussed on page 52 of COMAL Today #5. The three page article goes into detail about how it works.

ADVENTURES COMING SOON

Question: Please inform me when any text adventures written in COMAL are available. Is the language such that it can handle parsers of variable ability? From the low of 2 four letter words to a high of complex paragraphs?

Answer: COMAL Adventure games will be released soon. We are testing several of them now. Watch for future announcements in COMAL Today. COMAL can handle parsing of text of any length. The more you parse, the more complex the program.

SPRITES

Question: Sprites usage question: How would the "Up, Up and away" program on

More ►

WHY THE : IN :=

Question: What is the meaning of the colon between a letter variable and the "="?

A:=255 // notice the colon

Answer: There are two kinds of 'equal'. You can check to see if something is equal or not, or you can ASSIGN it to be equal. The first leaves the variable unchanged, while the second changes it. You may use the same equal sign (=) for both cases. However, COMAL will insert the colon in front of each equal sign that is performing an assignment.

A:=255 // assignment
IF A=255 THEN // just checking

GOSUB ---> PROC

Question: It appears to me that if there is a PROCedure of a certain name, one can place the name on a line by itself, and the program will GOSUB to it, in the BASIC sense. Right?

Answer: Yes. But the COMAL method is greatly improved over the primitive BASIC GOSUB. With COMAL, you also can pass parameters or call it from direct mode. And calling it by name is much better than by a number!

BASIC: GOSUB20000:REM PAUSE
COMAL: PAUSE

MEMORY MAP?

Question: Is there a COMAL memory map?

Answer: See COMAL Today #6.

HOW TO DELETE A LINE

Question: I have just recently received a COMAL disk. The language is wonderful. I am currently in college studying Computer Science, and am learning COBOL. Next semester I am supposed to take Pascal. By using COMAL now, I will be very familiar with the structure of Pascal. In studying some of the sample programs, I have figured out how to write some simple things, but I have one major problem at present. I do not know how to delete a line. Typing the line number alone, does not do it. I have resorted to renumbering the lines and then commenting out (//) all the line numbers at the end. Please explain how to delete a line. Thank you. - Paul Winslow, Millis, MA

Answer: To delete one line use the DEL command:

DEL 50

The example above would delete line 50. The DEL command also can delete a whole series of lines at once. Use the same method of specifying lines as with the LIST command:

DEL 100-200 // delete lines 100-200
DEL -400 // delete lines 1-400
DEL 500- // delete lines 500-9999

In COMAL 2.0 you also can delete procedures and functions by name:

DEL PRINTOUT

Finally, in COMAL 2.0 you can specify several "blocks" of lines with one DEL command separated by commas:

DEL 50,600,830-880

More ►

page 71 of the 64's User Guide, (except for the sprite data) be coded in COMAL? Is the so called "seam" anything to worry about? What is the turtle's sprite data like for showing others what the turtle looks like, but may not have a copy of COMAL?

Answer: COMAL has keywords to control sprites. Absolutely no PEEK or POKE is necessary. The data statements that create the sprite image can be used unchanged, but a final "64th" byte is required to specify the type of sprite image (0=hi res, 1=multi color). You do not have to worry about the "seam", or any other complications involved when trying to manipulate sprites from BASIC. COMAL takes care of everything for you. COMAL's turtle image is held by sprite number 7. This is normally a calculated image, since not only does it change as the turtle's direction changes, but it can be 10 different sizes, controlled by the keyword TURTLESIZE. There was an interesting article and program on page 33 of COMAL Today #5 about how you can change the turtle image to be anything you would like.

FREE MEMORY

Question: I read an overall review of COMAL in the November 1984 issue of *RUN*. It stated that COMAL has only 9902 bytes free. That doesn't sound like much, until you realize COMAL is so powerful, what it can do in 10K, BASIC needs about 18K to accomplish the same thing! Is this true?

Answer: Yes. COMAL reserves a section of memory for up to 64 sprite images. In BASIC, this memory would have to be allocated specially by the program. COMAL also reserves memory for either a

hi-res or multi-color graphics screen; a BASIC program would have to create this area itself. Finally, COMAL also is much more efficient in storing its program. It tokenizes not only keywords, but variable names as well. A 63 character variable name takes up only one byte of program memory. Finally, COMAL 0.14 now has been expanded to 11,838 bytes free, as explained on page 19 of COMAL Today #5.

SPEEDSCRIPT CONVERT?

Question: On *Today Disk* #8, please give me explicit instructions on using "seq'to'speed". On the same disk, how does one use the program, "view'sprites"? I tried LOAD and ENTER in both memory versions of COMAL 0.14.

Answer: The Speedscript conversion programs are explained on page 56 of COMAL Today #8. You should LOAD the program from disk into COMAL 0.14. And, you caught a problem. The COMAL 2.0 version of "view'sprites" program is on the COMAL 0.14 side of the disk. Thanks for bringing this to our attention. We will try to make a 0.14 version of the program for next issue.

SPACE REQUIRED

Question: Why is a space required between keywords and variables?

```
PRINT CHR$(154) //correct
PRINTCHR$(154) // not correct
```

Answer: COMAL allows variable names of up to 78 characters long. Thus, PRINTCHR could very well be a variable name. Because of this valuable feature, always remember to include a space between keywords and variable names.

More ►

CURSOR COMMANDS

Question: On page 110 of the *Cartridge Tutorial Binder* there are directions for using the CURSOR command (or statement). When entering

CURSOR 15,30

(or any other combination of row and column) the cursor will go to the row designated but NOT to the column. Can you explain this? Could it be a fault in my C64? - Anthony Conca, Warwick, RI

Answer: The CURSOR command works perfectly in a running program. However, if you issue a CURSOR command from direct mode, remember, that COMAL always returns to the first position in the next line after executing the command. Thus the CURSOR command was executed correctly. But immediately afterward, COMAL put the cursor at the beginning of the next line. This is correct - not what you expected perhaps, but correct.

FRAME

Question: What is the FRAME command used for? According to my COMAL pocket reference card, it is used to "set up a screen window", but when I have tried to use it in my programs, in either text mode or graphics mode, it doesn't seem to do anything. - John F Eldredge, Nashville, TN

Answer: In COMAL 0.14 the FRAME command does create a window - on the graphics screen. Drawing is only allowed inside this window (which is the entire screen by default). The command is rarely used. This command is enhanced in COMAL 2.0 and named WINDOW.

Rodney McDaniel of Jonesboro, AR has these questions:

WHERE TO GET COMAL?

Question: Is COMAL only available from the COMAL Users Group? On what machines is COMAL presently available and for what price?

Answer: C64 COMAL is also available from many Commodore User Groups. A complete list of COMAL vendors is printed in this issue.

COMAL COMING UP

Question: Will COMAL be available for any of the newly released computers?

Answer: Yes. As new computers come out, COMAL implementors work on versions for them. Soon we hope that COMAL will come with the computers.

HARD DISKS

Question: Will COMAL work with hard disks (such as a 10 Meg Winchester)?

Answer: Yes. We have a 20 Meg hard disk hooked up to our Zenith running IBM PC COMAL. It works wonderfully.

80 COLUMN SCREENS

Question: Will COMAL work on 80 column screens?

Answer: Yes. Most do. The Commodore 64 version is restricted to 40 columns by the computer, not by COMAL. A program to allow 80 column output on the C128 with COMAL 2.0 is included in this issue. ■

Letters

Dear Len- The enclosed video tape and diskette contains some "programming" that may be of interest to you. As a teacher, I have been using COMAL programs in a secondary school environment for the past couple of years. Since the advent of the cartridge version I have been able to develop a nifty technique for putting math lessons on videocassettes. These can be viewed alone or in conjunction with a C64.

I'd like to spread the word about what I am doing and thought that *COMAL Today* would be an appropriate vehicle to accomplish this. If you can find the time to view the tape and try the software you should have a pretty good idea of the concept I have in mind.

An instructor can use this software along with all the other features on the *COMAL Cartridge* to place math lessons on a videocassette using only the C64 and an inexpensive VCR. On playback, the student can view the videotape for content only, or can link up with a C64 so that he or she can attempt to duplicate or extend what appears on the video monitor. All that is required to switch back and forth is to press the play or stop button on the VCR. It is also possible to use the software without a VCR. Graph-Paper is an excellent tool for classroom demonstrations by a teacher or for experimenting with by students. - Garrett Hughes, Shelburne, VT

Hopefully next issue we will have his article explaining how this is done. It is a very interesting idea.

GET PEOPLE INTERESTED

Dear Sirs- I recently received COMAL 0.14. The language features are nice and well thought out. I am happy to be a "COMAL Programmer".

After trying to interest several people at our local university, the city's high school, and a local Commodore computer club, I was amazed and a little distressed at the lack of interest shown for COMAL. Admittedly, these people have developed a certain amount of C64/CBM BASIC expertise and invested time and money in their software inventories of BASIC and ML software. The opportunity to program and use programs of a superior language should have, however, swayed at least one of these educated and influential people.

Do you have any suggestions as to how COMAL can be presented and have people listen. Thank you for a fine language. - Rodney McDaniel, Jonesboro, AR

You are asking people to change, and change is often hard. However, COMAL has so many advantages, that once enough people start using it, the rest will follow. The Auto Run Demo Disk is a very nice way to show off COMAL in action. The Tutorial Disk and Bricks Tutorial disk are both good systems to show to educators. Then start the barrage of facts about COMAL. Wear them down! Once a few switch, the going gets easier! Finally, write to the 'big' magazines. Ask why no COMAL articles? The magazines will listen if enough of us write to them. It will be easier for you to present COMAL to your colleagues with COMAL articles from various national magazines.

More ►

RE-AWAKENED PROGRAMMER

Dear Sirs- COMAL has re-awakened my love of programming. I learned programming back in 1970, in Fortran with punch cards. I also learned ALGOL and a little known language called SNOBOL (text oriented). I missed the start of the micro revolution, being at the time, more interested in girls and swimming. Two years ago, I bought a Vic 20, then the C64 because the college where I taught math needed an introduction level, part time teacher. I subscribe to *Run, Ahoy, Compute's Gazette*, *TPUG Magazine*, and the *Transactor*.

Before receiving COMAL, I didn't feel like *doing anything* with the C64, and used it only as a word processor. Now, my leisure time is shared between COMAL and my 13 month-old daughter!

COMAL Today is full of flavor, the flavor that only a magazine whose articles are supplied by fervent users can have. Please keep up the excellent work! I know that someone there must be doing a lot of work, collecting and retyping all those articles. I particularly enjoy the non glossy paper you use (it's easier on the eyes) and the fact that it isn't filled by advertisements. Mostly, I like reading *COMAL Today* because it is informative: like the language itself, each article is aimed at teaching the amateur programmer (like myself) how to use new tools to solve his or her problems. I'm getting sick of other magazines offering a quarter page article, mostly useless blabber, to present a four page listing (in the back of the magazine) made up of 98% of DATA statements. I want to know how it works!

I have read somewhere in *COMAL Today* that COMAL is a "3-pass interpreter" and "run-time compiler". What does that mean exactly? Where can I find an explanation of what goes on inside COMAL? Thanks. - Louis Philippe Thouin, Quebec, Canada

PASS 1: Syntax checking of each line as it is entered.

PASS 2: Structure check. Makes sure all program structures are proper (each WHILE ends with ENDWHILE, etc.)

PASS 3: All branching within the program is converted to absolute address branching. For example, a procedure call is converted to the address that the procedure begins at. This allows the program to run faster.

Those are the 3-passes. Run-time compiler is a term used to refer to the code stored internally in a semi-compiled state, completely tokenized and compressed. But it still requires the COMAL interpreter to be present in order to run. The articles listed below cover several aspects of "what goes on inside COMAL":

COMAL 2.0 Internal Structure, COMAL Today #9, page 50

COMAL 2.0 Token Table, COMAL Today #9, page 54

Show The COMAL 2.0 Name Table, COMAL Today #9, page 68

C64 COMAL, COMAL Today #8, page 28

Using the INTERRUPT Command, COMAL Today #8, page 62

COMAL 2.0 External Procedures, COMAL Today #7, page 27

Batch Files From Memory, COMAL Today #7, page 32

COMAL 2.0 Memory Map, COMAL Today #6, page 22 ■

Price Protection

Every *COMAL Today* subscriber is protected. If you buy any COMAL item from us, and the price goes down within 1 month, you are entitled to a credit for the difference. Now you can buy with confidence. We wonder how many other computer companies will be willing to follow our lead with this policy. Note: special prices offered at a show, conference, or similar event do not count as a price drop.

DISCOUNTS ON NEARLY EVERYTHING

All *COMAL Today* subscribers now automatically get a discount on nearly everything they buy from us. We will continue our special quantity prices to schools and groups as well.

THE NEW PRICE STRUCTURE

Keep it simple. What good is a discount if it takes you an hour to calculate? Effective December 1, 1985, this is how our subscriber discount works:

- \$1 off each CheatSheet keyboard overlay
- \$1 off each back issue of *COMAL Today*
- \$2 off every book (now includes all books)
only \$4.95 for optional matching disk
- \$4 off *COMALite* shirts
- \$4 off each *COMAL 2.0 Deluxe Cartridge Pak*
- All other disks are \$9.75 each
- \$10 off each *McPen Lightpen*
- \$15 off the set of 4 *Cartridge Demo Disks*

CREDITS AVAILABLE - PROTECTION PLAN

To claim your credit you must return your original invoice showing your purchase at the higher price. Circle the items and clearly mark the credit amount. The credit can be applied to any future order. Notice: to claim credit on the special on books, you must have ordered two or three of the specified titles at the same time. ■

Ask Someone Who Knows

If you enjoy Jim Strasma's many books, and his articles in this and other magazines, you'll be glad he also edits his own highly-acclaimed computer magazine, now in its sixth year of continuous publication. Written just for owners of Commodore's many computers, each *Midnite Software Gazette* contains hundreds of brief, honest reviews.

Midnite also features timely Commodore news, hints and articles, all organized for instant reference, and never a wasted word. Whether you are just beginning or a long-time hobbyist, each issue will help you and your computer to work together effectively.

A six issue annual subscription is \$23. To subscribe, or request a sample issue, just write:

MIDNITE SOFTWARE GAZETTE
P.O. Box 1747
Champaign, IL 61820

You'll be glad you did!

C.A.S.E.
commodore association of the south east

presents

THE COMMODORE SHOW

at the

Opryland Hotel

Nashville, Tennessee
April 26th & 27th, 9:00 a.m. to 5:00 p.m.

NATIONAL COMMODORE SPEAKERS!
VENDORS AND DISPLAYS!

SHOW SPECIALS AND DOOR PRIZES!

SEE INNOVATIONS AVAILABLE FOR THE COMMODORE MARKET!
THE ONLY COMMODORE CONFERENCE IN THE SOUTH EAST!
USER GROUP ORIENTED!

2-DAY REGISTRATION FEE: \$10.00 (\$7.50 before April 1)
Tickets are available at all affiliated clubs,
or for more information, contact:

C.A.S.E. • P.O. BOX 110386 • NASHVILLE, TN 37222
(615) 834-5689 • (615) 834-2073 • (205) 854-3496

How to Type in COMAL Programs

Line numbers are irrelevant to a running COMAL program. COMAL only provides line numbers for your benefit in editing the program. Thus most magazines do not use line numbers when listing a COMAL program. It is up to YOU to provide the line numbers. But of course, **COMAL can do it for you quite easily.** Just follow these steps to type in a COMAL program:

- 1) Enter command: NEW
- 2) Enter command: AUTO
- 3) Type in the program
- 4) When done:
Version 0.14: Hit <return> key twice
Version 2.0 : Hit <STOP> key

Remember - use unshifted letters throughout entering the program. If letters are capitalized in the listing it does not mean to use SHIFT with those letters. They are capitalized merely to be easy to read. The only place to use SHIFTED letters is inside quotes. Also, you don't have to type leading spaces in a line. They are listed only to emphasize structures. You **DO** have to type a space between COMAL words in the program.

LONG PROGRAM LINES: We are continuing to print COMAL TODAY with two columns per page, printed with 40 characters maximum

per line. This makes it easiest to read. However, some program listings have program lines that extend beyond the 40 character limit. Unless we use a smaller type, we list these lines in the same manner that COMAL uses when listing long lines on a 40 column screen. We simply break the line, and continue it on the next line, indenting it properly to keep the program structures obvious. These are called wrap lines. To draw your attention to these continued lines we add a //wrap line comment to the end of the line. Whenever you see this make sure you type both lines as one continuous program line! The following example includes a line with more than 40 characters that we must list on two lines, but you must type in as one long program line:

```
if current$name$<>"finish" then print'la  
bel(current$name$,phone$) //wrap line
```

If you type in this long program line as two shorter program lines, COMAL will not object (although sometimes it will)! But, the program will not work unless it is entered as one long line. The procedure name PRINT'LABEL is split onto two lines in the listing, but the //wrap line draws your attention to this fact.■

COMAL Today welcomes contributions of articles, manuscripts and programs which would be of interest to readers. All manuscripts and articles sent to COMAL Today will be treated as unconditionally assigned for publication and copyright purposes to COMAL Users Group, U.S.A., Limited and is subject to the Editor's unrestricted right to edit and to comment editorially. Programs developed and submitted by authors remain their property, with the exception that COMAL Users Group, U.S.A., Limited reserves the right to reprint the materials, based on that published in COMAL Today, in future publications. There will be no remuneration for any contributed manuscripts, articles or programs. These terms may be varied only upon the prior written agreement of the Editor and COMAL Users Group, U.S.A., Limited. Interested authors should contact the Editor for further information. All articles and programs should be sent to COMAL Users Group, U.S.A., Limited, 6041 Monona Drive, Madison, WI 53716. Authors of articles, manuscripts and programs warrant that all materials submitted are original materials with full ownership rights resident in said authors. No portion of this magazine may be reproduced in any form without written permission from the publisher. Local Users Groups may reprint material from this issue if credit is given to COMAL Today and the author. Entire contents copyright (c) 1985 COMAL Users Group, U.S.A., Limited. The opinions expressed in contributed articles are not necessarily those of COMAL Users Group, U.S.A., Limited. Although accuracy is a major objective, COMAL Users Group, U.S.A., Limited cannot assume liability for article/program errors.

Please note these trademarks: Commodore 64, CBM of Commodore Electronics Ltd; PET, Easy Script of Commodore Business Machines, Inc; Calvin the COMAL Turtle, Captain COMAL, COMAL Today of COMAL Users Group, U.S.A., Limited; Buscard, PaperClip of Batteries Included; CP/M of Digital Research; Z-80 of Zilog; IBM of International Business Machines; Apple of Apple Computer Inc; PlayNET of PlayNet Inc; Compute!, Compute!'s Gazette of Compute! Publications, Inc. Sorry if we missed any others.

Order Form

Name: _____ SUBSCRIBER NUMBER: _____ Mar. 1986
(required for reduced prices-except new subs)
Street: _____ Pay by check/MoneyOrder in US Dollars
Canada Postal US Dollar Money Order is OK
City/St/Zip: _____ VISA / MasterCard print card#/exp date: _____
VISA/MC #: _____ exp date: _____ Signature: _____

Only Price List/Subscriber price-Item Description (all disks Commodore 1541 format) Prices subject to change

SYSTEMS:

(two disks may be supplied as a double sided disk)

- [] _____ \$98.95/\$94.95 Deluxe Cartridge Pak (2 books/1 disk/1 cartridge)-(shipping add \$4)
- [] _____ \$19.95/\$17.95 Programmers Paradise Pak (Fastloaded Comal 0.14/400 pgs info)(shipping add \$2)
- [] _____ \$5/\$5 option only with Paradise Pak - (includes Tutorial disk, Best of Disk, Auto Run Demos)
- [] _____ \$350/\$350 IBM Denmark PC COMAL (Danish manual/COMAL Handbook)-(shipping add \$5)

SUBSCRIPTIONS:

- [] _____ COMAL Today newsletter-> How many issues? _____ Start with: 6 7 8 9 10 11 12<-Circle one
(\$14.95 first 6; \$2 each added issue; >>> Canada add \$1 per issue; >>> overseas add \$5 per issue)
- [] _____ \$14.95/\$12.95 First 4 issues COMAL Today spiral bound (shipping add \$2)
- [] _____ \$3.95/\$2.95 COMAL Today backissue: circle issues wanted-> 5 6 7 8 9 10 -(ship add \$1 each)
- [] _____ TODAY DISK subscription >>>>>>> How many disks? _____ Start with disk# _____
(\$49.95/\$39.95 for first 6 disks - \$5 each added disk)-(no extra shipping charge)

BOOKS: (optional matching disks are \$14.95/\$4.95)-(>>>>Canada shipping add \$1 more per book<<<<)

- [] _____ \$18.95/\$16.95 COMAL Handbook - optional disk [] (shipping add \$3)
- [] _____ \$17.95/\$15.95 Starting With COMAL (matching disk not available yet)-(shipping add \$3)
- [] _____ \$19.95/\$17.95 Foundations With COMAL - optional disk [] (shipping add \$3)
- [] _____ \$28.95/\$26.95 Structured Programming With COMAL - optional disk [] (shipping add \$3)
- [] _____ \$20.95/\$18.95 Beginning COMAL - optional disk [] (shipping add \$3)
- [] _____ \$17.95/\$15.95 C64 Graphics With COMAL 0.14 - optional disk [] (shipping add \$3)
- [] _____ \$6.95/\$4.95 COMAL From A To Z (part of Paradise Pak)-(shipping add \$2)
- [] _____ \$14.95/\$12.95 Captain COMAL Gets Organized (includes disk)-(shipping add \$2)
- [] _____ \$6.95/\$4.95 COMAL Workbook (perfect companion to Tutorial Disk)-(shipping add \$2)
- [] _____ \$14.95/\$12.95 COMAL Library of Functions & Procedures (includes disk)-(shipping add \$2)
- [] _____ \$14.95/\$12.95 Graphics Primer (includes disk) (for COMAL 0.14)-(shipping add \$2)
- [] _____ \$6.95/\$4.95 Cartridge Graphics & Sound (part of Deluxe Cartridge Pak)-(shipping add \$2)
- [] _____ \$19.95/\$17.95 COMAL 2.0 Packages (disk includes C64SYMB & Monitor)-(shipping add \$2)
- [] _____ \$19.95/\$17.95 Packages Library (with disk)-(shipping add \$2)
- [] _____ \$19.95/\$17.95 Cartridge Tutorial Binder (with disk)(part of Deluxe Cart Pak)(shipping add \$3)
- [] _____ \$14.95/\$12.95 COMAL Quick 0.14 (fastloaded) with Utility Disk #2 & book (shipping add \$2)

DISKS:

- [] _____ \$14.95/\$9.75 Font Disk for COMAL 0.14 & 2.0 (40 different fonts plus support programs)
- [] _____ \$94.05/\$89.95 19 Disk Set (about 1000 programs for COMAL 0.14)-(shipping add \$3)
- [] _____ \$14.95/\$9.75 Best of COMAL 0.14 (new version - single side of disk)
- [] _____ \$14.95/\$9.75 Auto RUN Demo and Tutorial Disk (perfect with COMAL Workbook)
- [] _____ \$14.95/\$9.75 Bricks Tutorials (2 sided BEGINNERS disk)
- [] _____ \$14.95/\$9.75 Utility Disk #1 for COMAL 0.14
- [] _____ \$14.95/\$9.75 TODAY Disk-Circle disks wanted: 1 2 3 4 5 6 7 8 9 10 (see subscription above)
- [] _____ \$10/\$9.75 User Group Disks-Circle disks wanted: 1 2 3 4 5 6 7 8 9 10 11
- [] _____ \$14.95/\$9.75 Cartridge Demo Disks-Circle disks wanted: 1 2 3 4 (or see next line)
- [] _____ \$29.95/\$14.95 Set of all Cartridge Demo Disks (includes #1 #2 #3 and #4)
- [] _____ \$14.95/\$9.75 Specialty Disks (0.14 & 2.0): [] Games Disk [] Typing Disk [] Modem Disk

OTHER:

- [] _____ OTHER: _____
- [] _____ \$3.95/\$2.95 Keyboard Overlay for C64 COMAL 0.14 - Cheatsheet (shipping add \$1)
- [] _____ \$49.95/\$39.95 McPen Light Pen (with COMAL 2.0 Demo Disk)-(shipping add \$3)
- [] _____ \$9.95/\$5.95 Royal Blue COMALite Shirt-circle ADULT size: S / M / L / XL -(shipping add \$2)
- [] _____ \$170.35 School COMAL Package - 12 different books and 12 different disks (shipping add \$7.20)

Total _____ + _____ Shipping (minimum \$2 per order) = Total Paid US\$ _____ (WI add 5% sales tax)
Mail To: COMAL Users Group USA, 6041 Monona Drive, Madison, WI 53716 or call 608-222-4432

Subscriber's Specials

This issue's subscriber-only specials are good only through **April 30, 1986**. If you are a current subscriber you **MUST** include your subscriber number with your order, or we will not honor the special prices. Your subscriber number is printed on the newsletter mailing label and all our invoices. New subscribers get these prices automatically and a number will be assigned with the order.

BIG BOOKS SPECIAL

COMAL Handbook, *Beginning COMAL*, and *Foundations In Computer Studies With COMAL* are favorite books. If you don't have them yet, now is your chance. Take any two of them for \$26.90, or all three for only \$37.85. Normal shipping charges apply.

BACK ISSUES ONLY \$1.50 EACH

Order anything at all, and get COMAL TODAY back issues for only \$1.50 each. No extra shipping charge on UPS packages. First Class (Canada) add \$1 extra per issue.

FREE DISK OFFER

- *** Order Library of Functions & Procedures for \$12.95, get Utilities Disk #1 free.
- *** Order User Group Disk #10 for \$9.75 and get User Group Disk #11 free.
- *** Order Today Disk #2 for \$9.75 and get Today Disk #1 free.

UNITED STATES COMMODORE COUNCIL

JOIN AMERICA'S LARGEST COMMODORE USERS SUPPORT GROUP

INITIAL DUES \$25.00/RENEWAL \$20.00

BENEFITS

- * ACCESS TO THOUSANDS OF PUBLIC
- * DOMAIN PROGRAMS.
- * ONE YEAR SUBSCRIPTION TO
- * USCC COMMAND PERFORMANCE.
- * CONSUMER ASSISTANCE.
- * TECHNICAL ASSISTANCE.
- * FREE 35+ UTILITY PROGRAMS

USCC COMMAND PERFORMANCE

PUBLISHED BI-MONTHLY
PRODUCT EVALUATIONS
SOFTWARE REVIEWS
HOW TO FEATURES
PROGRAMMING INFORMATION
LATEST PRODUCT INFORMATION
INFORMATIVE FACTS

BONUS ** FREE VIEWTRON STARTER KIT & 1 HOUR ON-LINE TIME

* MAIL YOUR APPLICATION TO: USCC, P.O. BOX 2310, ROSEBURG, OR 97470 *

COMAL 0.14 ERROR MESSAGES

Error Message

- 0 Format error
- 1 Syntax error
- 2 Type conflict
- 3 Function argument error
- 4 Statement too long or complicated
- 5 System error
- 6 Name too long
- 7 Bracket error
- 8 Overflow
- 9 Error in structured statement
- 10 Error in goto statement
- 11 Stack overflow
- 12 Unknown variable
- 13 Procedure param error
- 14 Index/Param error
- 15 Substring error
- 16 Command, array, substring, procedure error
- 17 Index Error
- 18 Illegal number of indices
- 19 String assignment error
- 20 Function argument error
- 21 Not implemented
- 22 ZONE value incorrect
- 23 STEP = 0
- 24 Array redefined
- 25 Dimension error
- 26 CASE error
- 27 End of data
- 28 File already open
- 29 File input error
- 30 End-of-File
- 31 File not open
- 32 CON not possible
- 33 Error in print using
- 34 division by zero
- 35 program not prepassed
- 36 File not found
- 38 not input file
- 39 Device not present
- 40 Not output file
- 41 String not dimensioned
- 42 Local variable error
- 52 Too many names
- 53 Function value not returned
- 54 Not a statement
- 55 Not a command or simple statement
- 56 ',' expected
- 57 Number out of range
- 58 Expression expected
- 59 Not implemented
- 60 Operand expected
- 91 User error #1
- 92 User error #2
- 100 Graphic not active
- 101 Illegal color
- 102 Illegal plot coordinates

Filename Conventions

Suffixed	Prefixed	Meaning
NAME	NAME	COMAL program file
NAME.L	LST.NAME	Program listed to disk
NAME.PROC	PROC.NAME	PROC listed to disk
NAME.FUNC	FUNC.NAME	FUNC listed to disk
NAME.DAT	DAT.NAME	Data file
NAME.TXT	TXT.NAME	Text file
NAME.DOC	DOC.NAME	Documentation file
	EXT.NAME	External PROC/FUNC
	SHAP.NAME	Sprite shape file
	FONT.NAME	COMAL font file
	FONT.MC.NAME	Multicolor font file
	SET.NAME	Basic type font file
	PKG.NAME	Package file
	BAT.NAME	Batch file
	SNG.NAME	Song file
	HRG.NAME	Color COMAL picture
NAME.HRG		Black/White bitmap
	CRG.NAME	Compacted color pix
NAME.CRG		Compacted B/W bitmap
	ICON.NAME	Print Shop type Icon
	SCRN.NAME	Text Screen File
	POP.NAME	Mergeable Popover
NAME.POP		Program with Pop

Rebecca Daisywheel...



© 1985 G. RAYMOND EDDY

IF YOUR LABEL SAYS
LAST ISSUE: 11
YOU MUST RENEW NOW.
USE ORDER FORM INSIDE

BULK RATE
U.S. POSTAGE
PAID
MADISON, WI
PERMIT 2981

COMAL TODAY
COMAL USERS GROUP, U.S.A., LIMITED
6041 Monona Drive, Madison, WI 53716